



# فَنجَانٌ مِّنَ الِ - Cup Of Java



لغة بَرَمَجَة  
الكائنات الموجهة

المهندس : مؤنس قشقوش

نحف - الجليل الاعلى 2013

# فنجان من الـ

# JAVA

لغة برمجة الكائنات الموجهة

المهندس : مؤنس قشقوش

نحف – الجليل الاعلى 2013

kmoness@gmail.com

## الإهداء

إلى أحبائي

إلى أمي الحبيبة مريم رمز التضحية ونكران الذات التي علمتني حب الناس وعلمتني العمل في صمتٍ بعيداً عن الأضواء وحب الظهور .

وإلى أبي حسين الذي غرس في نفسي حب العلم منذ الصغر، وقد تعلمت منه أن القناعة لا تعني صغر الهمة، وأن التواضع لا يعني الذل.

وإلى زوجتي الغالية نهال التي هيئت لي المناخ المناسب لإستكمال كتابة هذا الكتاب فهي خير مشجع لي على هذا الطريق .

والى اولادي شهد مريم ومحمد الذين احبهم حباً عظيماً .

ولمن علمني حرفاً وكان عوناً لي على الطريق و لكل محب للخير .

اللهم تقبل منا هذا العمل واجعله خالصاً لوجهك الكريم وأجرنا يا رحيم. وأسأل من قرأه فإستفاد ألا ينسنا من صالح دُعائه وصادق رجائه.

وإلى كل هؤلاء الذين احبهم أقدم هذا الكتاب.

المهندس: مؤنس قشقوش

**فهرست المواضيع:**

صفحة 5	✓ مقدمة
صفحة 8	✓ لماذا لغة الجافا - ميزات الجافا
صفحة 11	✓ مكونات لغة الجافا
صفحة 16	✓ عمليات إسنادية
صفحة 27	✓ الفئات والكائنات بلغة الجافا
صفحة 28	✓ عمليات بنائية - Constructors
صفحة 31	✓ التحميل الزائد للبيانات - Methods OverLoading
صفحة 32	✓ متغيرات ثابتة Static Data Members
صفحة 32	✓ الدوال الثابتة Static Methods
صفحة 33	✓ أسئلة
صفحة 38	✓ نقل قيم من عملية بانية الى عملية بانية أخرى
صفحة 40	✓ فئة تحوي على موجه من فئة أخرى Composition
صفحة 41	✓ مجمع المهملات [ Garbage Collection ]
صفحة 45	✓ مصفوفة كائنات Arrays & Collections
صفحة 48	✓ تمارين - كائنات ومصفوفة كائنات
صفحة 64	✓ الوراثة Inheritance
صفحة 70	✓ وراثة بعدة أدوار (طبقات) (Chain Inheritance)
صفحة 74	✓ المخطط الهرمي للوراثة (Inheritance Hierarchies)
صفحة 76	✓ متغير من نوع Protected
صفحة 80	✓ Overriding and Hiding Methods (إخفاء وإعادة كتابة)
صفحة 84	✓ تعدد الأشكال - Polymorphism
صفحة 89	✓ الدالة الوهمية (virtual function)
صفحة 91	✓ تحويل موجه الأساسية Base الى موجه من المشتقة Derived
صفحة 93	✓ الفئة Object
صفحة 97	✓ معالجة الاستثناءات Exception Handling
صفحة 103	✓ أنواع الاستثناءات Exception Types
صفحة 109	✓ الواجهات - Interfaces
صفحة 123	✓ وراثة متعددة للواجهات
صفحة 141	✓ أسئلة عن الواجهات
صفحة 155	✓ ArrayList, Collections And Polymorphism
صفحة 165	✓ أسئلة متنوعة وحلول



# مقدمة

نسخة تجريبية - ليست للنشر

## مقدمة

التطور الكبير في تقنية صناعات الحاسبات الآلية وانتشارها في جميع مجالات الحياة المختلفة ، واستخداماتها المتعددة في شتى المجالات، فإنه أصبح إلزاما علينا معرفة هذه الحاسبات وكيفية التعامل معها والاستفادة منها لأنها توفر الجهد والوقت وتنجز كثير من الأعمال بدقه متناهية بالإضافة إلى قدراتها الكبيرة في الإحتفاظ بالبيانات. من الطرق الشائعة للاستفادة من القدرات الكبيرة للحاسبات هو: بناء البرامج التي تقوم بحل كثير من المشكلات توفيراً للجهد والوقت. في هذه الوحدة سوف نلقي الضوء على ماهية برنامج الحاسوب وكذلك انواع لغات البرمجة المختلفة. ثم بعد ذلك نبين أهمية مهنة البرمجة وصناعة البرمجيات.

### البرنامج :

البرنامج هو عبارة عن مجموعة من التعليمات تعطى للحاسب للقيام بعمل ما مثل حساب مجموعات مختلفة ، حساب المتوسط الحسابي ، حساب مضروب عدد معين. والبرنامج هو الذي يحدد للحاسب كيفية التعامل مع البيانات للحصول على النتائج المطلوبة. والبرنامج يكتب بواسطة المبرمج الذي يفهم المشكلة ويقترح الحل وينفذه بواسطة كتابة برنامج في إحدى لغات البرمجة .

والبرمجيات (Software): هي التي تسهل للمستخدم إستخدام المكونات المادية (Hardware) بكفاءة وراحة عالية ويمكن تقسيم البرمجيات إلى ثلاثة أنواع رئيسية وهي :

برامج التشغيل Operating system: مثل النوافذ (windows, Dos ,Linux) .  
برامج التطبيقات Application Programs: مثل البرامج (Excel, PowerPoint..) .  
لغات البرمجة Programming Language: وهي برامج تساعد في إنشاء البرامج المختلفة برامج التطبيقات تتعامل مباشرة مع المكونات المادية للحاسوب.

تقسم البرمجيات الى ثلاثة أقسام :

- لغة الآلة Machine languages  
وهي اللغة الوحيدة التي يفهمها الحاسب ويستطيع التعامل معها . وهذه اللغة تعتبر لغة خاصة لكل حاسب وقد تختلف من حاسب إلى آخر وهي تعتمد على المكونات المادية للحاسب نفسه ، ولغة الآلة تتكون من مجموعة أرقام من 0 , 1 التي تعطي تعليمات للحاسب للقيام بمعظم العمليات الأساسية واحدة بعد الأخرى.

- لغات التجميع Assembly languages  
هي لغة تستخدم اختصارات مبيرة من اللغة الإنجليزية لتعبر بها عن العمليات الأولية (التي يقوم بها الحاسب مثل إضافة add حفظ save طرح sub الخ..)

Load     A

Add     B

Save     C

- لغات المستوى العالي High level languages  
وهذه اللغات كتبت بحيث تستخدم بعض الكلمات الإنجليزية العادية بنفس معانيها حيث يقوم كل أمر فيها بتنفيذ العديد من الواجبات، وهذه اللغات كسابقتها تحتاج إلى مترجمات Compilers التي تقوم بتحويل التعليمات (الأوامر) إلى لغة الآلة ، وهذه اللغات تستخدم العلاقات والعوامل الرياضية المتعارف عليها. مثال ذلك:

$$S = A + B * C$$

- وهذه اللغات تعتبر سهلة ومرغوبة من وجهة نظر المبرمجين بالمقارنة بلغات التجميع ولغة الآلة وذلك لسهولة كتابتها وفهمها وحل المشاكل باستخدامها، ومن أمثلة هذه اللغات لغة VB, Fortran , C++ , C , JAVA وغيرهم.

## أهمية مهنة البرمجة

من المعلوم أن الذي يقوم بكتابة البرامج لحل المشكلات الكثيرة والمعقدة هم المبرمجون ولا يمكن الاستغناء عنهم بحال من الأحوال لأن دورهم مهم وحيوي وتكثر الحاجة لهم في شتى المجالات وذلك لعمل الآتي:

- كتابة برامج وبناء الأنظمة المختلفة لحل المشاكل وتبسيط التعامل مع الحاسب .
- المسؤولية الكاملة عن إصلاح ما يحدث من أعطال أو حل المشاكل التي تحدثها الأنظمة المختلفة.
- بناء واجهة المستخدم المختلفة في كثير من اللغات والتطبيقات .
- بناء نظم التشغيل المختلفة مثل Unix, Windows وغيرها. فمثلا تستخدم لغة C بناء نظام التشغيل Unix.
- برامج المواجهة المختلفة في الأنظمة المختلفة الرقمية و التماثلية.

## لماذا لغة الجافا - ميزات الجافا:

الجافا لغة سهلة التعلم Simple ، حيث أنها بنيت باستخدام C++ ، لذلك فقد أخذت الكثير من لغة C++ ، ومن هنا أي مبرمج C++ يمكنه تعلم هذه اللغة واساسياتها في عده أيام فقط ، وحتى اذا لم تكن مبرمج C++ ، فسوف تتعلم هذه اللغة بسهولة أيضا ، فليس بها الكثير من التعقيد كما بعض اللغات مثل المؤشرات والتعامل اليدوي مع الذاكرة ، فقط أحجز ما تريد ولا تقلق بكفيه تحريره ، فسوف يتولي هذا الأمر مللمم النفايات Garbage Collector ، وكما قال مخترع هذه اللغة جيمس غوسلنق أنه أخذ أفضل خصائص لغة C++ وأسهلها ووضعها في لغته ، وترك الخصائص الأخرى المزعجة للمبتدئ مثل المؤشرات والوارثه المتعدده واضاف ما هو أفضل.

الجافا لغة Full Object Oriented ، جميعنا نعلم أن البرمجة الموجهه للكائنات oop ، هي أحد أهم متطلبات البرمجة في هذا الزمان ، ونظرا لأن البرمجة تسهل البرنامج كثيرا من حيث الفهم والاستيعاب ، وتجعله قابل لاعاده الإستخدام Reusability وسهل الصيانه والعديد من الأمور الأخرى . ولو أردت أن تكتب برنامج لطباعه Hello World سوف تستخدم هذه المفاهيم أيضا مثل Class ، والوراثه Inheritance أيضا!

في جافا تستطيع برمجه برامج شبكيه موزعه networked/distributed environments، أغلب البرامج الحديثه وخاصه في بعض المنشآت ، تتطلب أن يستطيع أكثر من مستخدم Client الدخول للبرنامج في نفس اللحظه ، وأستدعاء دوال معينه موجوده في السيرفر ، وهنا جافا وفرت لنا طاقم رائع للتعامل مع البرامج الشبكيه ، حيث يمكننا عمل اي برنامج نريده مثل شات ماسنجر وبكل سهوله أيضا . أيضا مع دعم لتقنيه RMI نستطيع استدعاء دالة موجوده في كائن موجود في جهاز بعيد ، أيضا دعم لتقنيه COBRA وهي نفس الخاصيه ولكن استدعاء دوال مكتوبه بلغات أخرى مثل C++ .

جافا لغه صممت من الأساس على مبدأ المحموليه Portable ، أي أن البرنامج المكتوب فيها تحت ويندوز يعمل بلا أي مشاكل في أي نظام تشغيل آخر يحتوي على أله جافا الافتراضيه JVM طبعا الفكره VM تعود الى مخترع لغه باسكال وصاحب الأسبقيه في هذا المجال وهو نيكولاث ويرث ) ، لذلك شعار جافا هو WORA ، أي Write Once , Run Anywhere وهذه الخاصيه من أهم الأسباب لاختيارك للغه ما ، فمثلا من الخصائص المزعجه في C++ أنها تعتبر حجم int هو على حسب المترجم الذي تم ترجمه به ، ففي C++ Trbuo حجم int هو 2 بايت ، اما في المترجمات الأخرى 4 بايت ، وهو أمر مزعج جدا في حال أردت نقل برنامج لمنصه أخرى واعاده ترجمته فقط تكون هناك مشاكل في التشغيل . أما في جافا جميع أنواع البيانات لها حجم ثابت بغض النظر عن البيئه المستخدمه.

جافا تدعم مفهوم تعدد المسارات Multithreaded ، تعدد المسارات أحد اهم المفاهيم في البرامج الكبيره ، مثلا برنامج متصفح انترنت ، اذا لم يكن هناك Thread فسوف تنتظر الى أن ينتهي تحميل الصوره الأولى ، والثانيه والثالثه وهكذا حتى تستطيع كتابه موقع جديد لفتحه ! وكل البرامج الموجوده الان تدعم هذا المفهوم من محرر النصوص word الى برامج التشات والماسنجر.

وفي جافا تستطيع كتابه برنامج Multithreaded ببضعه أسطر ، وهذا ما يجعلها اللغه المفضله وخصوصا لدى السيرفرات التي دعم أكثر من عميل في نفس اللحظه

Java Is Strong in Server Side .

جافا تدعم الاف من المكتبات المساعده ، من مكتبات رياضييه Math ، تعامل مع أعداد ضخمة جدا Big Number ، تعامل مع دوال توليد أرقام عشوائيه ، مكتبات خاصه بالتشفير وطرق التوقيع الرقمي Cryptography ، مكتبات خاصه للتعامل مع الجرافيك 2D , 3D Graphics ، مكتبات خاصه للتعامل مع الواجهات GUI ، وو الكثير من المكتبات التي تساعدك على أداء وظيفتك بكل سهوله.

Java & Internet ، عن طريق الأبلت تستطيع عمل تطبيقات تعمل في صفحه الويب ، وهنا تستطيع كتابه أي برنامج تريد ، ولكن تدخل خاصيه الأمن هنا ، فلن يستطيع \*\*\*\*\* كتابه أي معلومات أو قرائه ملف من جهاز العميل . وبرامج الأبلت منتشرة بكثرة وأشهرها هي غرف الحوار والدرشه Chat ، أيضا لها تستخدم في Intranet اي في الشبكة المحليه لشركة ما ، حيث يكتب برنامج ، ويقوم الموظفون بالدخول الى هذا البرنامج والتعامل معه

## مكونات لغة الجافا

### Components of Java programming languages

هذه الوحدة تبحث في مكونات لغة الجافا التي تتكون من متغيرات (Variables) ثوابت (Constants) عمليات حسابية (arithmetic operation) العمليات المنطقية (logic) العلاقات (Relational) النصوص (strings) وغيرها .

#### مثال 1 :

برنامج يطبع جملة ترحيبية

```
public class Welcome
{
    public static void main( string args [] )
    {
        System.out.println( "Welcome to Java Programming!" );
    }
}
```

هنالك تشابه كبير بين لغة C# ولغة الجافا , نكتب البرنامج والدوال دائماً داخل فئات , وهنالك الدالة الرئيسية main التي تنفذ أولاً من قبل المترجم . الأمر System.out.println يقوم بطباعة نص على الشاشة .

الكلمات المحجوزة في لغة الجافا :

abstract	finally	public	boolean	float
return	break	for	short	byte
if	static	case	implements	super
catch	import	switch	char	instanceof
synchronized	class	int	this	continue
interface	throw	default	long	throws
native	transient	double	new	true
else	null	try	extends	package
void	false	private	volatile	final
protected	while			



مثال 2 :

برنامج يطبع جملة ترحيبية بأكثر من سطر

```
public class Welcome
{
    public static void main( string args [] )
    {
        System.out.print( "Welcome to " );
        System.out.println( "Java Programming!" );
    }
}
```

الأمر System.out.print يقوم بالطباعة بدون ان ينزل سطر (يبقى المؤشر على نفس السطر) أما الأمر الثاني فإنه يطبع الجملة وبعد الطباعة يقوم بإنزال المؤشر الى بداية سطر جديد.

بعض الحروف الخاصة في الطباعة :

يمكن إدخال الأحرف التالية في جمل الطباعة كل له تأثير معين

الحرف	الوصف
\n	سطر جديد , المؤشر يظهر في بداية السطر التالي
\t	مسافة أفقية , تحريك المؤشر مسافة معينة في نفس السطر

إظهار نص في صندوق الحوار

بالرغم من إظهار النص السابق في نافذة الأوامر، إلا أن الكثير من تطبيقات الجافا تستخدم صناديق الحوار لإظهار النصوص بدلاً من نافذة الأوامر، معظم البرامج وبخاصة متصفحات الانترنت مثل (internet explorer) تستخدم صناديق الحوار في كثير من التطبيقات .

وصناديق الحوار هي عبارة عن نافذة يتم إظهار الرسائل المهمة الموجهة للمستخدم فيها وتسمى message Dialog.

**مثال 4 :**

```
import javax.swing.JOptionPane;
public class Welcome4
{
    public static void main(string args[])
    {
        JOptionPane.showMessageDialog(null,"Welcome\nto\nJava\nProgramming");
        System.exit(0); // نهاية البرنامج
    }
}
```



يظهر نفس النص السابق في صندوق حوار يسمى (message Dialog). وأحد عناصر القوة في لغة الجافا هو احتوائها على العديد من الكائنات الجاهزة التي يمكن للمبرمجين إعادة استخدامها ثانية بدلا من إنشائها من البداية. جملة تعليق في سطر واحد تبين الجزء من البرنامج الذي يشير إلى استدعاء كائن معين من حزمة الامتداد باستخدام import.

تقسم جمل import الى أقسام :

جمل import للحزم الرئيسية (Java)

جمل import لحزم الإمتداد (Javax)

جمل import للحزم الخاصة (Deitel)

السطر : `import javax.swing.JOptionPane`

المترجم يستخدم جملة لكي يتم تعريف وتحميل الكائنات المستخدمة في لغة الجافا , وفي هذه الحالة استدعاء وضم للكائن `JOptionPane` .

وعند استخدام الكائنات من API فإن المترجم يتأكد من استخدامها بالصورة الصحيحة. وجملة `import` تساعد المترجم في أن يحدد ويجد الكائن ولذلك فإنه عند استخدام أي كائن من الجافا API يجب تعريف الحزمة التي تحوي هذا الكائن .

يمكن معرفة بعض المعلومات عن الحزم والكائنات الموجودة في لغة الجافا من الموقع

<http://Java.sun.com/j2se/1.3/docs/api/index.html>

إستدعاء الدالة `show.Message.Dialog` المعرفة داخل الكائن `JOptionPane` , وهذه الدالة تتلقى قيمتين الأولى مكان الذي يظهر به الصندوق (عادة يكون `null`) , والثاني النص المراد إظهاره في الصندوق.

### مثال 5:

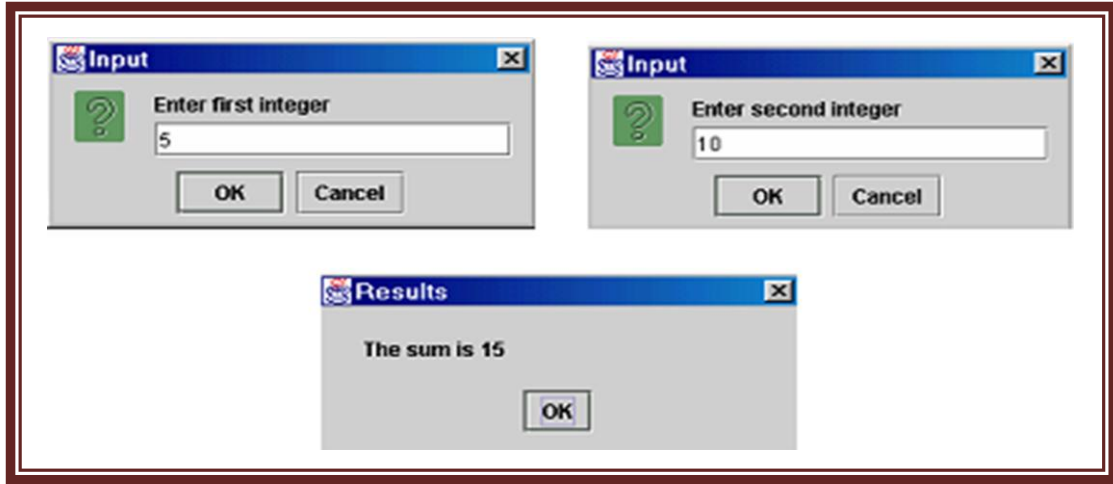
```
import javax.swing.JOptionPane;

public class Addition
{
    public static void main( String args[] )
    {
        String fNum;
        String sNum;
        int num1;
        int num2;
        int sum;

        fNum=JOptionPane.showInputDialog("Enter first integer");
        sNum=JOptionPane.showInputDialog("Enter second integer");
        num1 = Integer.parseInt(fNum);
        num2 = Integer.parseInt(sNum);

        sum = num1 + num2;
        JOptionPane.showMessageDialog(null,"The sum is " + sum, "Results",
        JOptionPane.PLAIN_MESSAGE );

        System.exit( 0 );    // terminate application
    }
}
```



استقبال عددين داخل المتغيرين num1 و num1 جمعهم داخل المتغير sum ثم طباعة النتيجة مع إظهار جملة توضيحية .

### أنماط المتغيرات – variable types

النوع (Type)	الحجم بال- Byte	المدى (Range)	ملاحظات
boolean	1 byte	True / False	قيمة منطقية
char	2 byte	to FFFF...	متغير يأخذ رمزاً واحداً فقط
byte	1 byte	-128 to +127	قيمة صحيحة
short	2 byte	-32768 to +32767	
int	4 byte	-2147483648 to +2147483647	
long	8 byte	-9,223,372,036,854,775,808 to +9,223,372,036,854,775,807	
float	4 byte	-3.40292347 E+38 to +3.40292347 E+38	قيمة حقيقية , بها فاصلة عشرية
double	8 byte	-1.79769313488231570 E+308 to +1.79769313488231570 E+308	

عمليات إسنادية :

العملية	ماهية العملية
+	جمع - Addition
-	طرح - Subtraction
*	ضرب - Multiplication
/	قسمة - Division
%	الباقى من القسمة - Modulus
++	زيادة قيمة المتغير بواحد - Pre or Post Increment
--	نقصان قيمة المتغير بواحد - Pre or Post Decrement

علاقات منطقية (مقارنات) - Relational operators

الإشارات المستخدمة في العلاقات المنطقية تقارن بين قيمتين وتعيد قيمة منطقية , True أو False

العلاقات المنطقية (مقارنات)		العوامل المنطقية Logical operators	
العملية	ماهية العملية	العملية	ماهية العملية
$X > Y$	هل X أكبر من Y	&&	وأيضاً - And
$X < Y$	هل X أصغر من Y		أو - Or
$X \geq Y$	هل X أكبر أو مساوٍ لـ Y	!	عكس - Not
$X \leq Y$	هل X أصغر أو مساوٍ لـ Y		
$X == Y$	هل X مساوٍ لـ Y		
$X != Y$	هل X يختلف عن Y		

العوامل التي تعمل على بت bit operators :

لغة الجافا تمكن المبرمج من استخدام عوامل (عمليات وإشارات معينة) على مستوى ال bit وهو أصغر وحدة تخزين في الحاسوب عبارة عن خلية واحدة تحوي قيمة واحدة 0 أو 1 . ماهية هذه العوامل انها تقارن بت مع بت اخر بالترتيب . البت الأول من المتغير الأول يقارن مع البت الأول من المتغير الثاني , البت الثاني من المتغير الأول يقارن مع البت الثاني من المتغير الثاني وهكذا .

العملية	ماهية العملية
&	AND - على مستوى ال bit
	OR - على مستوى ال bit
^	XOR - على مستوى ال bit
~	NOT - على مستوى ال bit
<<	Left Shift - إزاحة يسارية على مستوى ال bit
>>	Right Shift - إزاحة يمينية على مستوى ال bit

## أدوات التحكم في لغة الجافا

### أمر الشرط : if - else المبنى العام :

```
if ()
{
    ----أوامر للتنفيذ
    ----
}
else
{
    ----أوامر للتنفيذ
    ----
}
```

### الأمر switch :

```
switch (switch-expression)
{
    case value1: أوامر للتنفيذ; break;
    case value2: أوامر للتنفيذ; break;
    ...
    case valueN: أوامر للتنفيذ; break;
    default : أوامر للتنفيذ; break;
}
```

### الحلقة while التكرارية :

```
; قيمة بداية
While ( شرط )
{
    ----أوامر للتنفيذ
    ----
}
```

### الحلقة do-while التكرارية :

```
do
{
    ----أوامر للتنفيذ
    ----
} while ( شرط إستمرار الحلقة ) ;
```

حلقة for التكرارية :

```

( زيادة او نقصان المتغير ; شرط استمرار الحلقة ; قيمة بداية للمتغير )
for
{
    ----
    ----  أوامر للتنفيذ
    ----
}

```

مثال 1:	مثال 2:
<pre> for (int j=1 ; j &lt;= 9; j++ ) {     System.out.print ( " "+j);     System.out.println ( " " ); } </pre>	<pre> float i; for (i=0.1f; i&lt;=1.0f ; i= i+0.1f) {     sum += i ;     System.out.println("summation="+sum); } </pre>

حلقة for المتداخلة :

```

( زيادة او نقصان المتغير ; شرط استمرار الحلقة ; قيمة بداية للمتغير )
for
{
    ( زيادة او نقصان المتغير ; شرط استمرار الحلقة ; قيمة بداية للمتغير )
    {
        ----
        ----  أوامر للتنفيذ
        ----
    }
}

```

مثال 1:
<p>يطبع جدول الضرب من 1 حتى 100 كل عشر اعداد على سطر</p> <pre> for (int i=1 ; i &lt;= 10; i++ ) {     for (int j=1 ; j &lt;= 10; j++ )         System.out.print ( " " + i*j);     System.out.println(); } </pre> <p>حلقة تطبع حاصل ضرب العددين //</p> <p>بعد طباعة كل سطر المؤشر يقفز سطر جديد //</p>

جمل break و continue :

تستعمل هذه الجمل عندما يراد تغيير المسار الطبيعي للبرنامج فمثلا عندما تستخدم جملة break داخل بناء جملة for, while, do/while , switch تسبب الخروج منها فوراً ويستمر تنفيذ باقي جمل البرنامج التي تلي بناء الجملة والاستخدام الشائع لجملة break هو للهروب مبكراً من تنفيذ حلقة أو لإهمال تنفيذ باقي جملة switch.



## تعريف المصفوفة ذات البعد الواحد :

المصفوفات هي عبارة عن مواقع يتم تخزين البيانات فيها لمدة مؤقتة (طيلة فترة تنفيذ البرنامج فقط ) ، وعند تعريف المصفوفة وإنشاءها يتم حجز عدد محدد من المواقع المتجاورة في الذاكرة لتخزين البيانات فيها ، حيث يتم الوصول للبيانات المخزنة في هذه المواقع عن طريق اسم المصفوفة ورقم الموقع (index).

والغاية من استخدام المصفوفات هي تخزين عدد غير محدد من القيم تحت اسم واحد فقط (اسم المصفوفة ) دون الحاجة إلى تخزين كل قيمة في متغير منفصل.

```
int array1[ ] = new int[9];
int [ ] array1 = new int[9];
```

هنالك طريقتان لتعريف مصفوفة من الأعداد الصحيحة بمقدار 9 خلايا.

نستطيع بشكل إختياري أن نحدد للمصفوفة قيماً ابتدائية يتم تحديدها عند تعريف المصفوفة ، وإذا لم نحدد للمصفوفة قيماً ابتدائية فإنه يتم تخزين القيمة التلقائية (Default Value) لنوع المصفوفة وذلك عند حجز المواقع لها. والقيم التلقائية للأنواع هي كما يلي:

int, byte, short, long	→	0	
double, float	→	0.0	
char	→	\u0000	فراغ
string	→	null	
Boolean	→	false	

ويمكن تحديد القيم الابتدائية للمصفوفة بالطريقة التالية :

```
int ages[ ] = {20, 18, 34, 42, 28};
```

من خلال هذه الجملة قمنا بتعريف مصفوفة اسمها ages ، وخزنت فيها قيماً ابتدائية ، حيث سيتم حجز مواقع على عدد هذه القيم الابتدائية .

## تعريف المصفوفة الثنائية - Two-Dimensional Arrays

في لغة جافا يمكن تعريف مصفوفات ذات أكثر من بعد واحد ، وكمثال على ذلك :  
تعريف المصفوفات ذات البعدين . ونستطيع القول بأن المصفوفة ذات البعدين هي  
عبارة عن جدول يحتوي على صفوف وأعمدة ،  
تعريف المصفوفات:

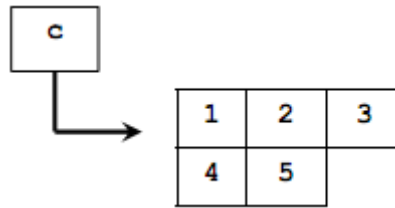
```
int b[ ][ ];
b = new int[3][4];
```

والمثال التالي يوضح كيفية تعريف مصفوفة ذات بعدين وإعطائها قيما ابتدائية :  

```
int b[ ][ ] = { { 1, 2 }, { 3, 4 } };
```

يمكن للمصفوفة أن تحوي على عدد مختلف من الأعمدة , الصف الأول يحتوي على  
ثلاث أعمدة والصف الثاني يحتوي على عامودين .  

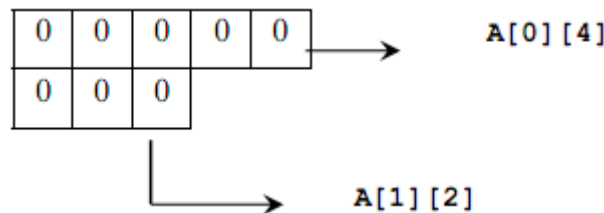
```
int c[ ][ ] = { { 1, 2, 3}, { 4, 5 } };
```



والمثال التالي يبين عملية تعريف مصفوفة وحجز مواقع لها بحيث يحتوي كل صف  
من صفوف هذه المصفوفة على عدد مختلف من الأعمدة :

```
int a[ ][ ];
a = new int[2][ ];
a[0] = new int[5];
a[1] = new int[3];
```

الشكل التالي يوضح كيف تم حجز المصفوفة الثنائية,



## الفئة Math :

تشبه جدا الفئة الموجودة في لغة C#.

الطريقة	وصف	مثال
abs(x)	القيمة المطلقة ل X	Math.abs(6.2) → 6.2 Math.abs(-4.2) → 4.2
ceil (x)	تقريب X الى أقل عدد صحيح ليس أقل من X (الصحيح الأكبر)	Math.ceil(5.1) → 6 Math.ceil(-5.1) → -5
floor (x)	تقريب X الى أكبر عدد صحيح ليس أكبر من X (الصحيح الأصغر)	Math.floor(5.1) → 5 Math.floor(-5.1) → -6
max(x,y)	أكبر قيمة من X و Y.	Math.max(7,6) → 7
min(x,y)	أقل قيمة من X و Y.	Math.min(-7,-8) → -8
pow(x,y)	X مرفوع للقوة Y.	Math.pow(6,2) → 36
sqrt(x)	الجذر التربيعي ل X.	Math.sqrt(9) → 3
random()	تكوين رقم عشوائي بين الصفر والواحد.	Math.random() → 0.24421

مثال 1 للتطبيق

```

public class UseMath
{
    public static void main( string args[])
    {
        System.out.println("The square root of 100 = " + Math.sqrt(100));
        System.out.println("The absolute value of 40 = " + Math.abs(40));
        System.out.println("The absolute value of -60 = " + Math.abs(-60));
        System.out.println("The absolute value of 10 = " + Math.abs(10));
        System.out.println("6 to the power 2 = " + Math.pow(6,2));
    }
}

```

مثال 2 للتطبيق

```

public class Example2
{
    public static void main(string args[])
    {
        int face ;
        for (int i = 1; i<=5 ; i++)
        {
            face = 1 + (int)(Math.random()*6);
            System.out.println("The Face in Try "+i+" is "+face);
        }
    }
}

```

مثال 3 للتطبيق

إستقبال عددين وطباعة المجموع

```
import java.util.Scanner;

public class Addition
{
    public static void main( string args[] )
    {
        Scanner input = new Scanner( System.in );

        int number1;
        int number2;
        int sum;

        System.out.print( "Enter first integer: " );
        number1 = input.nextInt();
        System.out.print( "Enter second integer: ");
        number2 = input.nextInt();

        sum = number1 + number2;
        System.out.printf( "Sum is %d\n", sum );
    }
}
```

## النصوص - strings :

تعريف نص :

```
string str2;  
string str2 = "My Friend";
```

نص :											
0	1	2	3	4	5	6	7	8	9	10	11
'M'	'y'		'F'	'r'	'i'	'e'	'n'	'd'			

str2[3] = ?

str2[2]= ?

str2[14]= ?

مميزات النص + والدوال المساعدة :

الدالة	ماهية الدالة
<u>length()</u>	تعيد الدالة طول النص .
<u>s.indexOf(t)</u>	ترجع موقع أول مكان توجد فيه t داخل السلسلة الرمزية s.
<u>s.indexOf(t,i)</u>	ترجع موقع أول مكان توجد فيه t داخل السلسلة الرمزية s بعد الموقع i.
<u>s.indexOf(c)</u>	ترجع موقع أول مكان توجد فيه الحرف المخزن في المتغير c داخل السلسلة الرمزية s.
<u>s.indexOf(c, i)</u>	ترجع موقع أول مكان توجد فيه الحرف المخزن في المتغير c داخل السلسلة الرمزية s بعد الموقع i.
<u>s.lastIndexOf(c)</u>	ترجع موقع آخر مكان توجد فيه الحرف المخزن في المتغير c داخل السلسلة الرمزية s.
<u>s.lastIndexOf(t)</u>	ترجع موقع آخر مكان توجد فيه السلسلة الرمزية t داخل السلسلة الرمزية s.
<u>s.compareTo(t)</u>	تقوم الطريقة بمقارنة السلسلة الرمزية s مع السلسلة الرمزية t وتعيد رقم سالب إذا كانت s اقل من t وتعيد صفر إذا كانت s تساوي t وتعيد رقم موجب إذا كانت s أكبر من t.
<u>s.equals(t)</u>	تعيد True إذا كانت s مساوية t.
<u>s.startsWith(t)</u>	تعيد True إذا كان s يبدأ بالسلسلة الرمزية t.
<u>s.startsWith(t, i)</u>	تعيد True إذا كانت السلسلة الرمزية t موجودة في s بدءاً من الموقع i.
<u>s.endsWith(t)</u>	تعيد True إذا كان s تنتهي بالسلسلة الرمزية t.
<u>s.charAt(i)</u>	ترجع الحرف الموجود في الموقع i داخل السلسلة الرمزية s.

الدالة	ماهيّة الدالة
s.charAt(i)	ترجع الحرف الموجود في الموقع i داخل السلسلة الرمزية s.
s.substring(i)	ترجع جزء من السلسلة الرمزية s بدءاً من الموقع i وحتى النهاية.
s.substring(i , j)	ترجع جزء من السلسلة الرمزية s بدءاً من الموقع i وحتى الموقع j-1.
s.toLowerCase()	إنشاء سلسلة رمزية جديدة تحتوي كل ما في السلسلة الرمزية s بعد تحويل كل الحروف الى حروف صغيرة .
s.toUpperCase()	إنشاء سلسلة رمزية جديدة تحتوي كل ما في السلسلة الرمزية s بعد تحويل كل الحروف الى حروف كبيرة.
s.trim()	إنشاء سلسلة رمزية جديدة من السلسلة الرمزية s بعد الفراغات من البداية والنهاية .
s.replace(c1, c2)	إنشاء سلسلة رمزية جديدة من السلسلة الرمزية s بعد تبديل كل c1 بـ c2. وهما من النوع char.
s.matches(regexStr)	تعيد True إذا كانت السلسلة الرمزية regexStr تطابق السلسلة الرمزية s كاملة .
s.replaceAll(rStr, t)	إنشاء سلسلة رمزية جديدة بعد تبديل كل rStr بـ t .
s.split(rStr)	إنشاء مصفوفة تحتوي على أجزاء من السلسلة الرمزية s مقسمة حسب ظهور rStr .

#### مثال على إستعمال الدالة indexOf:

```
string first = "HeLlO My Friend";
char ch1 = "L";
string second = "My";
int plc1 = first.indexOf (ch1);
int plc2 = first.indexOf (second);
```

#### مثال على إستعمال الدالة Substring :

```
string str1 = "don't stop the war";
string str2 = str1.Substring (6);           → str2 = "stop the war";
string str3 = str1.Substring(11,14); → str3 = "the";
```

مثال على استعمال الدالة : replace

```
string secret = "always cheat and always steal";  
secret = secret.replace ("always" , "never");
```

مثال على استعمال الدالة : toUpperCase

```
string str1 = "always cheat and always steal";  
string str2 = str1.toUpperCase ();
```

مثال على استعمال الدالة : toLowerCase

```
string str1 = "ALWAYS cheat AND always steal";  
string str2 = str1.toLowerCase ();
```

مثال على استعمال الدالة : compareTo

```
string str1 = "peace";  
string str2 = "pears" ;  
int Res1 = str1.compareTo(str2); → Res1 = -10  
int Res1 = str2.compareTo(str1); → Res1 = +10  
int Res1 = str1.compareTo("peace"); → Res1 = 0
```

دمج بين نصوص (append) + :

توصل بين نصين , تدمج نصين بنص واحد ,

```
string str1 = "My";  
str1 = str1 + " Friend";
```

عمل ذاتي :

ماذا تفعل كل من البرامج التالية , يمكن ان تقع أخطاء صححها :

```
public class Advancedstring  
{  
    public static void main (string args[])  
    {  
        string st = "spring";  
        System.out.println (st + st);  
        System.out.println (st);  
        st = "hello " + st;  
        System.out.println (st);  
    }  
}
```



```
import java.util.Scanner;
public static void main (string args[])
{
    Scanner input = new Scanner( System.in );
    string st1, st2;

    System.out.println ( "Enter 2 strings" );
    st1 = input.next();
    st2 = input.next();

    if (st1.equals(st2))
        System.out.println(st1+" is equal to"+st2);
    else
        System.out.println("different strings");

    if (st2.compareTo(st1)==0)
        System.out.println (st2+" is equal to"+st1);
    else
        System.out.println ("different strings");
}
```

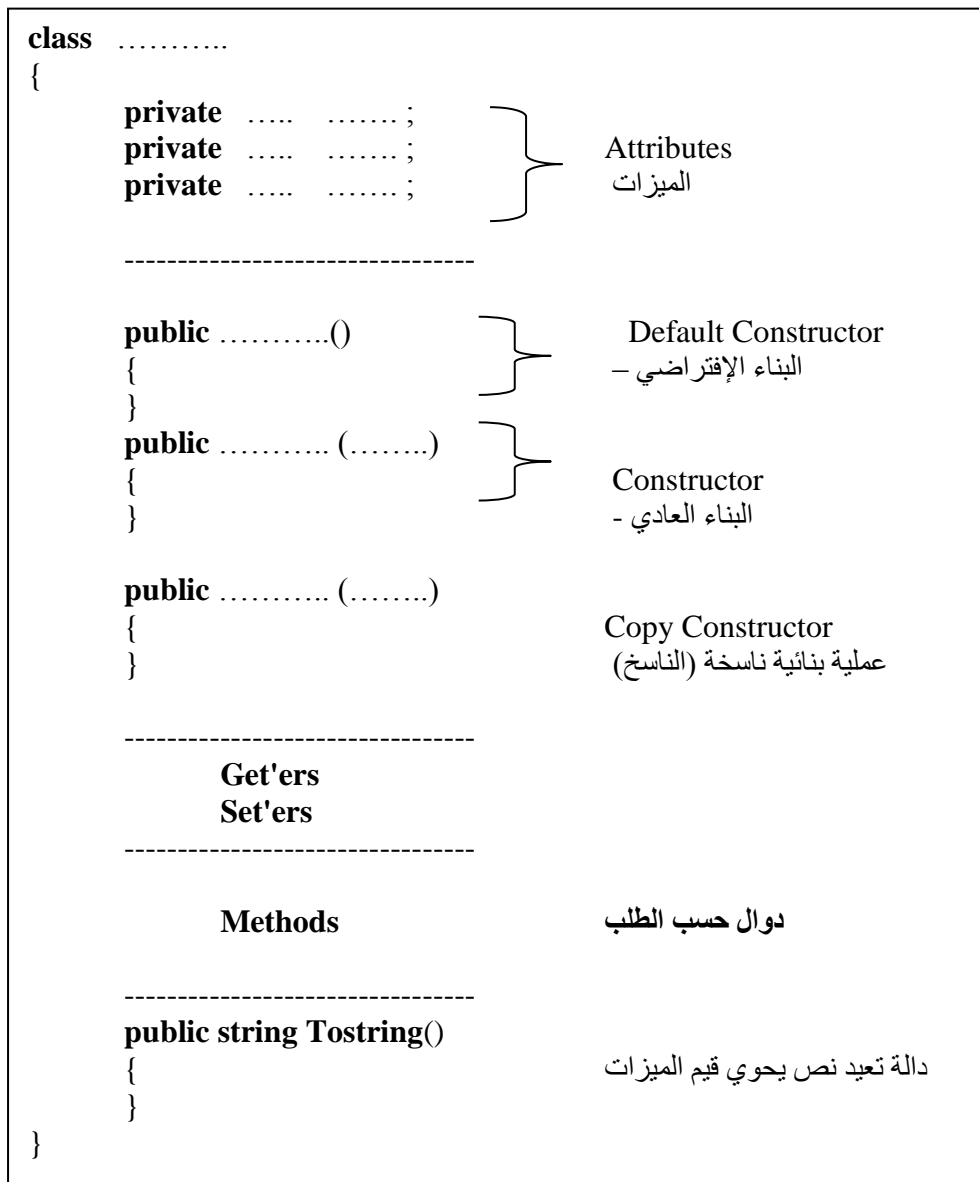
```
public static void main (string args[])
{
    string st = "abc";
    System.out.println ("first="+ st.length() );
    st = "abcdefg";
    System.out.println ("2nd="+ st.length());
}
```

```
public static void main (string args[])
{
    string st1 = "happy birthday";
    string st2, st3, st4;
    st2 = st1.substring(6);
    st3 = st1.substring(11);
    st4 = st2.substring(5);
    System.out.println ("st1= " + st1);
    System.out.println ("st2= " + st2);
    System.out.println ("st3= " + st3);
    System.out.println ("st4= " + st4);
}
```

الفئات والكائنات بلغة الجافا

## فئة – Class

## المبنى العام – Syntax



الهدف من البانيات هو إعطاء قيماً أولية لميزات الفئة, ويمكن للفئة الواحدة أن تحوي أكثر من بانية واحدة وهذا ما يسمى بالتحميل الزائد للبانيات (overloading).

نلاحظ كثرة استعمالنا للكلمات `public` , `static` وغيرها ولعلك أخذت فكرة على معنى كل كلمة من هذه الكلمات , فهي التي تحدد لنا مدى قدرتنا إلى الوصول إلى الكائن البرمجي سواء أكان متغيراً أم دالة ... , وهذه الكلمات سنلخصها في هذا الجدول لتعرف وقت إستعمال كل واحدة منها.

الكلمة	الإستعمال
<code>public</code>	الكائنات البرمجية معروفة في كل فئات المشروع
<code>private</code>	الكائنات البرمجية معروفة فقط على مستوى الفئة التي تنتمي إليها
<code>protected</code>	الكائنات البرمجية معروفة فقط على مستوى الفئة فقط والفئات التي ترثها (سنشرحها في الوراثة لاحقاً)
<code>Internal</code>	الكائنات البرمجية معروفة فقط على مستوى الفئات المنتمية إلى نفس <code>assembly</code> .

### المجمعات assemblies :

المجمع `assembly` عبارة عن ملف يضم كل ما يوجد في برنامجك من ملفات وفئات , ويكون إمتداده إما `exe` أو `dll` حسب نوع المشروع , إن كان تطبيقاً `application` كان إمتداد الأسمبلي `exe` وإن كان نوع المشروع مكتبة `Bibliothèque` كان إمتداده `dll`.

### إستعمال static :

المتغيرات والدوال التي تكون مسبقة بهذه الكلمة يمكننا إستعمالها من غير إستنساخ للفئة أي إستعمالها مباشرة بدون بناء كائن من نوع الفئة التي تتواجد بها الفئة .

### البانيات Constructors :

البناء هو عبارة عن دالة تحمل نفس إسم الفئة لكنها لا تعيد شيئاً , وهي أول جزء ينفذ عند إنشاء كائن من فئة معينة , دوره يتجلى في إعطاء قيم بدائية لمتغيرات الفئة كما سنرى في المثال التالي , وهو لا يحتاج على أن نعلن عن نوعه كما هو الحال مع الدوال العادية بل لا يقبل حق امتعمال `void` .  
إن كان البناء ينتظر قيمةً ليعطها لمتغيرات الفئة فإن الإعلان عن نسخة من هذه الفئة يكون كالتالي:

```
class object = new class(arg1,arg2,....,argN);
```

إن كانت الفئة تحتوي على مجموعة من البانيات فإن الإعلان عن كائن من هذه الفئة يستلزم التعامل مع البانيات , وإن لم يكن هنالك بناء واضح داخل الفئة فإنها تستعمل بناء افتراضي بدون متغيرات داخلية .

مثال :

```

class Point
{
    private int m_X;
    private int m_Y;
    //-----
    public Point ()
    {
    }
    public Point(int x,int y)
    {
        this.m_X = x;
        this.m_Y = y;
    }
    public Point(Point p)
    {
        this.m_X =p.getX();
        this.m_Y =p.getY();
    }
    //-----
    public void SetPoint(int X1,int Y1)
    {
        this.m_X = X1;
        this.m_Y = Y1;
    }
    //-----
    public void setX(int m_x)
    {
        this.m_X = m_x;
    }
    public int getX()
    {
        return this.m_X;
    }
    public void setY(int m_y)
    {
        this.m_Y = m_y;
    }
    public int getY()
    {
        return this.m_Y;
    }
    //-----
    @Override public string toString()
    {
        return ("X="+ m_X + " Y="+ m_Y);
    }
}

```

البناء الافتراضي

البناء العادي

البناء النسخ

دوال  
Get'ers Set'ersدالة toString  
تعيد نص هو قيم  
الميزات

```

class Circle
{
    private int m_X;
    private int m_Y;
    public static final float PI=3.14f;
    private int m_R;
    //-----
    public Circle(int m_X, int m_Y, int m_R)
    {
        this.m_X = m_X;
        this.m_Y = m_Y;
        this.m_R = m_R;
    }
    //-----
    public void Print()
    {
        System.out.println (m_X + " " + m_Y);
        System.out.println ("Radius: " + m_R);
        System.out.println ("Area: " + PI * Math.pow(m_R,2));
        System.out.println ("primeter: "+2* PI* m_R);
    }
    //-----
    public static void PrintPI()
    {
        System.out.println (PI);
    }
}

```

متغير ثابت القيمة  
له تبقى ثابتة دائماً.

- إستدعاء البناء يكون بشكل أوتوماتي , يُستدعى عند عملية إنشاء الكائن دائماً يوجد إستدعاء للبناء. عند كتابة البناء يجب الإهتمام بقواعد معينة :
- اسم البناء مشابه لإسم الفئة , ويكون public.
  - البناء لا يعيد قيم , وأيضاً ليس void.
  - البناء يمكنه أن يتلقى برامترات .
  - يمكن ومن المفضل إستعمال التحميل الزائد بالبيانات overloading.

### دالة رئيسية – main للمثال السابق

```

class App
{
    static void main()
    {
        Circle c = new Circle(100,100,10);
        c.Print();
        Circle c2 = new Circle(200,200,20);
        c2.Print();
        Circle.PrintPI();
    }
}

```

```

class App
{
    static void main()
    {
        Point p1 = new Point(30,30);
        System.out.println (p1.toString());

        Point p2 = new Point();
        p2.setX(10); p2.setY(40);
        System.out.println (p2.toString());

        Point p3 = new Point(p2);
        System.out.println (p3.toString());
    }
}

```

### التحميل الزائد للبيانات – Methods OverLoading

لغة الجافا تمكن المبرمج من إستعمال التحميل الزائد في البيانات والدوال, هذه الميزة تمكن إستعمال دوال بنفس الاسم داخل الفئة . اللغة تفرق بين الدوال ذات نفس الاسم بواسطة عدد البارامترات المستقبلية أو بواسطة نوع البارامترات , لا يمكن التفرق بين الدوال حسب القيمة المعادة أو اسم البارامترات المرسله .

```

class Employee
{
    private string last_name;
    private string first_name;
    private float salary_per_hour;
    private float working_hours;

    public void SetEmployee(string ln,string fn)
    {
        last_name = ln;
        first_name = fn;
    }
    public void SetEmployee(float sal,float hours)
    {
        salary_per_hour = sal;
        working_hours = hours;
    }
    public void Print()
    {
        System.out.println ("last name " + last_name);
        System.out.println ("first name " + first_name);
        System.out.println ("Salary " + CalcSalary());
    }
}

```

```

public static void main()
{
    Employee e1 = new Employee();
    e1.SetEmployee("Yosef","Salim");
    e1.Print();
    e1.SetEmployee(19.34f,160);    e1.Print();
}

```

**متغيرات ثابتة: Static Data Members**

المتغيرات الثابتة هي المتغيرات الخاصة بصنف أي أنها غير مرتبطة بأي كائن ينتمي لهذه الفئة. وتبدأ فترة حياة هذه المتغيرات عند عملية التحميل للفئة للذاكرة. أحياناً نريد أن نخزن قيمة مشتركة لجميع الكائنات من فئة معينة , السبب توفير بالذاكرة أو لحساب شيء معين. مثلاً نريد أن نعد عدد الكائنات من فئة معينة من بدء تنفيذ البرنامج في هذه الحالة نستعمل متغيرات ثابتة.

```
class Sample
{
    .....
    private static int Count;
    .....
}
```

**Static Data Members** هم المتغيرات المشتركة لجميع الكائنات من نفس الفئة. هو عضو للفئة وليس للكائن من الفئة. وتبدأ فترة حياة هذه المتغيرات عند تحميل الفئة للذاكرة وقبل بناء أي كائن من الفئة . التوجه للمتغيرات الثابتة لا تكون بمساعدة كائنات من الفئة أو الكائن **this** لأنه لا ينتمي للكائن , إنما بواسطة اسم الفئة.

**مثال :** نستدعي متغير ثابت بصورة مباشرة بدون بناء كائن من الفئة .

Sample.Count

**الدوال الثابتة Static Methods**

**Static Methods** تشبه **Static Data Members** هي دوال لا تنتمي للكائن إلا للفئة. **Static Methods** تكون جاهزة منذ لحظة تحميل الفئة للذاكرة. مثل **Static Members** أيضاً **Static Methods** تكون معرفة بواسطة الكلمة **Static** نستدعي الدوال الثابتة بصورة مباشرة بدون بناء كائن من الفئة.

```
class MathMethods
{
    public MathMethods()
    {
    }
    public static int Sum(int n1,int n2)
    {
        return n1+n2;
    }
    public static float Avg(int n1,int n2)
    {
        return (n1+n2)/2;
    }
}
```

```
static void main(string[] args)
{
    System.out.println (MathMethods.Sum(20,40));
    System.out.println (MathMethods.Avg(50,30));
}
```



## سؤال 1 :

```
public class DoubleA
{
    private double x, y;
    private boolean b;
    private static int counter = 0;
    public DoubleA()
    {
        this.x = 2;
        this.y = 2;
        counter++;
    }

    public DoubleA(double x, int y)
    {
        this.x = x;
        this.y = y;
    }

    public double getX() { return x; }
    public double getY() { return y; }
    public void setB(boolean b) { this.b=b;}
    public int getCounter() { return counter;}
    public double powXY()
    {
        if (b)
            return Math.pow (y,x);
        else
            return Math.pow (x,y);
    }
    @Override public string toString()
    {
        return "Double: x="+ x +" y= "+ y +" b="+ b +" counter="+counter;
    }
}
```

هل من الممكن أن نضيف دالة بنائية تستقبل بارامترين من نوع double؟ إذا كان الجواب نعم إشرح لماذا وأكتب الدالة , وإذا كان الجواب لا إشرح.

ما وظيفة الكلمة static التي تظهر قبل تعريف المتغير counter؟ وما أهمية المتغير من هذا النوع؟

أكتب دالة بنائية ناسخة في DoubleA?

- أكتب مقطع برنامج يحوي ما يلي:
- إبن كائن من الفئة DoubleA, مع قيم موجبة .
  - استدع الدالة powXY مرتين, مرة تنفذ فيها العملية في سطر if ومرة تنفذ فيها العملية في السطر else .
  - إبن كائن مطابق للكائن الأول .
  - إطبع الكائنين .
  - تتبع المقطع وصف وضعية الكائنات.

## سؤال 2 :

```
class A
{
    private int num, sn;
    private double f;
    private static int counter = 0;

    public A(int num, double f)
    {
        this.num = num;
        this.f = f;
        counter++; sn = counter;
    }
    public A(int num, int f)
    {
        this.num = num;
        this.f = f;
        counter++; sn = counter;
    }
    public double getF()
    {
        return f;
    }
    public double getNum()
    {
        return num;
    }
    public void set(int num)
    {
        this.num = num;
    }
    public void set(double f)
    {
        this.f = f;
    }
    public void set(int f)
    {
        this.f = f;
    }
    @Override public String toString()
    {
        return "Integer="+ num +" double="+ f +" sn="+ sn;
    }
}
```

```

class B
{
    private A memA;
    private string x;

    public B(int num, float f, string x)
    {
        memA = new A(num, f);
        this.x = x;
    }
    public B(A memA, string x)
    {
        this.memA =new A(memA.num, memA.f);
        this.x = x;
    }
    public string getX()
    {
        return x;
    }
    @Override public string toString()
    {
        return memA.ToString()+" string=" + x;
    }
}

```

في الفئة A وقع خطأ؟ ما هو الخطأ؟ إشرح ثم صحح.

في الفئة B وقع خطاين؟ بيّن ما هما؟ إشرح ثم صحح. إعط حلول للأخطاء أحدهما بدون أن نمحي.

ما وظيفة الكلمة static التي تظهر قبل تعريف المتغير counter ؟ وما وظيفة هذا المتغير في الفئة A ؟

تتبع المقطع وصف وضعية الكائنات:

```

static void main()
{
    A a1= new A(3,6);
    A a2= a1;
    a2.set(5);
    B b1= new B(a1, " I am a1");
    B b2= new B(6, 6, " I am a2");
    a2.set(6.5);
    System.out.println(a1);
    System.out.println(a2);
    System.out.println(b1);
    System.out.println(b2);
}

```

### سؤال 3 :

الفئة SubjectGrade, تمثل إسم الموضوع وعلامة الموضوع . الأسم subject من النمط string , العلامة grade من النمط int.

أمامك واجهة للفئة SubjectGrade (UML) :

وصف	عنوان الدالة
عملية بانية	SubjectGrade(string subject, int grade)
دالة تعيد إسم الموضوع	string getSubject()
دالة تعيد علامة الموضوع	int getGrade()
دالة تعدل إسم الموضوع	void setSubject(string subject)
دالة تعدل علامة الموضوع	void setGrade(int grade)
دالة تعيد نص به قيم الميزات : <grade> : <subject>	string toString()

أ. طبق الفئة SubjectGrade .

ب. تتبع المقطع التالي ؟ وصف وضعية الكائنات.

```
public class TestSubjectGrade
{
    public static void main()
    {
        SubjectGrade history= new SubjectGrade("History", 80);
        SubjectGrade cs= new SubjectGrade("Cs", 97);
        SubjectGrade math;
        cs.setGrade(history.getGrade());
        history.setGrade(99);
        math = cs;
        System.out.println("Math");
        System.out.println(cs);
        System.out.println(history);
        System.out.println(math);
    }
}
```

ج. كم كائن أنشئ من النمط SubjectGrade بعد تنفيذ المقطع ؟ إشرح.

د. أضف رقم تسلسلي (Id) لكل كائن أنشئ من النمط SubjectGrade, هذا العدد يكون مرتب من 1 حتى عدد الكائنات التي أنشئت ؟ كيف نفعل ذلك .  
هـ. أكتب عملية بنائية ناسخة للفئة . (للكائن المنسوخ نعطي عدداً تسلسلياً جديداً).

**سؤال 4:**

أمامك واجهة الفئة A (UML):

وصف	عنوان الدالة
عملية بائية تعطي قيمة أولية للميزة x حسب القسمة المرسلة.	A(int x)
عملية بائية ناسخة .	A(A a)
دالة تعيد الميزة x.	int getX()
دالة تعدل الميزة x.	void setX(int x)
دالة تعيد نص به قيم الميزات على النحو: x is: <x>	@Override string toString()

بالإعتماد على الفئة السابقة نكتب الفئات التالية:

<pre> public class B {     private A a;     private int y;      public void setA(int x)     {         this.a.setX(x);     }     @Override string ToString()     {         return a.ToString()+ "y             is:" + this.y;     } } </pre>	<pre> public class TestAB {     static void main()     {         A a1 = new A(6);         B b = new B(a1,11);         A a2 = new A(a1);         b. setA(12);         a1.setX(36);          System.out.println(a1);         System.out.println(a2);         System.out.println(b);     } } </pre>
---	--

(أ) أكتب الفئة A.

(ب) أنظر الى كل من العمليات البائية التي كتبت للفئة B, بين إذا كانت العمليات كتبت بالشكل

الصحيح . إذا كتبت بالشكل الصحيح ماذا تكون نتيجة تنفيذ الفئة TestAB إذا لم تكتب

بالشكل الصحيح بين ما هو الخطأ .

طريقة 1:	طريقة 2:
<pre> public B(A a, int y) {     this.a.setX(a.getX());     this.y = y; } </pre>	<pre> public B(A a, int y) {     this.a = a;     this.y = y; } </pre>

## نقل قيم من عملية بانية الى عملية بانية أخرى (من Ctor معين الى Ctor آخر)

من الممكن أن تتساعد عملية بانية معينة بعملية بانية أخرى لوضع قيم أولية للميزات الهدف هو الإستفادة بقدر المستطاع من بناء آخر، وعدم تكرار أسطر مكتوبة في الفئة بكلمات أخرى إستغلال مهام العملية البانية.

```
public class Employee2
{
    private string m_LastName;
    private string m_FirstName;
    private float m_SalaryPerHour;
    private float m_WorkingHours ;
    //-----
    public Employee2()
    {
        this("-----","-----");
        System.out.println("First Ctor");
    }
    public Employee2(string ln,string fn)
    {
        this(ln,fn,-1,-1) ;
        System.out.println("Second Ctor");
        m_LastName = ln;
        m_FirstName = fn;
    }
    public Employee2(string ln,string fn,float sal,float hours)
    {
        System.out.println("Third Ctor");

        m_SalaryPerHour = sal;
        m_WorkingHours = hours;
    }
    //-----
    public void SetEmployee(float sal,float hours)
    {
        m_SalaryPerHour = sal;
        m_WorkingHours = hours;
    }

    public void SetEmployee(string ln,string fn,float sal,float hours)
    {
        m_LastName = ln;
        m_FirstName = fn;
        m_SalaryPerHour = sal;
        m_WorkingHours = hours;
    }
}
```

```
//-----
private float CalcSalary()
{
    if(m_SalaryPerHour == -1 || m_WorkingHours == -1)    return -1;
    return m_SalaryPerHour * m_WorkingHours;
}
public void Print()
{
    System.out.println("frst="+ m_LastName + " Last=" + m_FirstName);
    float sal = CalcSalary();
    if(sal == -1)
        System.out.println("Missing salary information!");
    else
        System.out.println("Salary = " + sal);
}
}
```

مثال للدالة الرئيسية - main

```
public class App
{
    public static void main()
    {
        Employee2 e1 = new Employee2();
        e1.Print();

        System.out.println();

        Employee2 e2 = new Employee2("Sami","Yosif");
        e2.Print();

        System.out.println();

        Employee2 e3 = new Employee2("Salim","Yosri", 70, 140);
        e3.Print();
    }
}
```

تتبع البرنامج أعلا , وصف وضعية الكائنات .

#### Output:

Third Ctor  
Second Ctor  
frst=Sami Last=Yosif  
Missing salary information!

Third Ctor  
frst=null Last=null  
Salary = 9800.0

**فئة تحوي على موجه من فئة أخرى Composition**

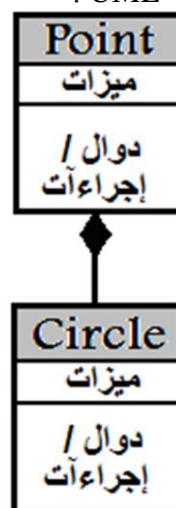
نغدير الفئة Circle – نكتبها بطريقة أخرى, الفئة Point – بدون تغيير :

الفئة Circle تحوي على موجه من فئة أخرى  
يوجد Reference – المعرّف من الفئة point .

```
class Circle
{
    private Point m_Center;
    private int m_Radius;

    //-----
    public Circle()
    {
        m_Center = new Point();
        m_Radius = 1;
    }
    public Circle(int x,int y,int r)
    {
        m_Center = new Point(x,y);
        m_Radius = r;
    }
    public Circle(Circle c)
    {
        m_Center = new Point(c.m_Center);
        m_Radius = c.m_Radius;
    }
    //-----
    public void Print()
    {
        m_Center.Print();
        System.out.println("Radius = {0}",m_Radius);
    }
}
```

مخطط UML :



مثال لدالة رئيسية :

```
static void main(string[] args)
{
    Circle c1 = new Circle(1,2,3);
    c1.Print();
    Circle c2 = new Circle();
    c2.Print();
    Circle c3 = new Circle(c1);
    c3.Print();
}
```

ماذا علينا ان نضيف للفئة حتى نستطيع تطبيق الدالة الرئيسية ؟  
يوجد نواقص بالفئة !



## أمثلة على إحتواء - أرسم مخطط UML للأمثلة التالية:

- 1) دكان (Store) من الممكن أن تحوي الفئة على كائن من نوع راديو (Radio) , كائن من نوع تلفاز (TV) , كائن من نوع ثلاجة (freezer) .
- 2) مصنع (Farm) من الممكن أن تحوي الفئة على كائن من نوع عامل (Employee) .
- 3) صف (ClassRoom) من الممكن أن تحوي الفئة على كائن من نوع طالب (student) وكل طالب يحوي الفئة على كائن من نوع موضوع (Subject).
- 4) مكتبة (Library) - تحوي الفئة على كائن من نوع كتاب (Book)
- 5) دكان موسيقي (DVDStore) تحوي الفئة على كائن من نوع قرص (CD)

## Garbage Collection :

### مجمع المهملات [ Garbage Collection ]

في لغات البرمجة الكائنية مثل السي والسي شارب cpp والجافا يوجد مفهوم ( نطاق العمل scope ) يقوم بتحديد عمر وفترة ظهور المتغير ويتحدد هذا النطاق في لغة الجافا باستخدام القوسين المعقوفين - الحاصرتين - { } - ويتحدد عمر المتغير بحاصرة النهاية بمعنى أنه يكون موجودا خلال الحاصرتين وحين تخرج من نطاق العمل سوف تشغل الذاكرة ولذلك في الجافا توجد ميزة تحرير الذاكرة .

ميزة تحرير الذاكرة باستخدام مجمع المهملات الخاص بالجافا [ Garbage Collector ]

هي الميزة التي يتم فيها تحرير الذاكرة من المتغيرات الغير المستخدمة حيث تخرج من نطاق العمل والذي يتحدد في لغات مثل cpp والسي والجافا بالحاصرتين [ { } ] والتي تشكل كتلة برمجية ويكون المتغير متاحا خلالها هذا بالنسبة للمتغيرات الأولية أما بالنسبة للكائنات التي تنشأ باستخدام الكلمة المحجوزة [ new ] كما في المثال التالي

```
{
    string alphabet = new string ("abcdefghijklmnopqrstuvwxyz");
}
```

نلاحظ هنا أن المتغير [alphabet] سيموت بإغلاق الخاصرة - نهاية نطاق العمل - بينما الكائن [string] الذي يدل عليه المعرف يظل في الذاكرة وهنا تأتي ميزة الجافا التي تميزها عن باقي اللغات وهي مجمع المهملات الذي يقوم بالبحث عن الكائنات التي تم إنشاؤها ولا يوجد لها معرف فيقوم بتحرير المساحة التي كانت محجوزة لها في الذاكرة ويجعلها متاحة لكائنات جديدة وكل هذا يحدث تلقائياً دون تدخل من المبرمج كما في لغة السي والسي شارب cpp وهذه ميزة خاصة بالجافا وبعبارة أخرى لا تشغل بالك ولا تقلق من تسرب الذاكرة فهناك من يعتني الأمر بدلاً عنك ودون أن تشعر تماماً مثل تناولك الطعام وتقوم المعدة بعملها دون أن تشعر بعملية الهضم

### الطريقة : finalize ()

ونأتي الآن إلى تسمية الطريقة التي تستدعى قبيل إنهاء الكائن بواسطة سلة المهملات الخاصة بالجافا هذه الطريقة أو الدالة تسمى ( finalize ) وهي تضمن أن يتم إنهاء الكائن وتحرير الذاكرة منه وعلى سبيل المثال تستخدم هذه الطريقة للتأكد من إغلاق الملف الذي يوجد به الكائن الذي خرج من نطاق العمل ولكي تضيف طريقة الإنهاء الى الفئة بحيث تستدعيها بيئة تشغيل الجافا ينبغي تعريف هذه الطريقة والشكل العام لاستدعاء الطريقة هو

```
protected void finalize()
{
    // finalization code here
}
```

ونلاحظ على هذه الصيغة أنها تبدأ بالكلمة المفتاحية المحجوزة [protected] لكي تمنع الوصول الى الفئة من خارجها

ومن لديه دراية سابقة بلغة ++C ربما يطرح سؤالاً عن شبيهه في تلك اللغة يقوم بالتخلص من الكائنات التي تخرج من نطاق العمل يسمى الهادم أو [destructor] الذي يختلف عن شبيهه في الجافا في أنه يستدعى قبيل خروج الكائن من نطاق العمل ولكن في الجافا [ نهج الإنهاء ] يعمل في الخلفية بدون تدخل من المبرمج ولكي تتضح الفكرة نضرب مثالا

مثال :

```
class FDemo
{
    int x;

    FDemo(int i)
    {
        x = i;
    }
    protected void finalize()
    {
        System.out.println("Finalizing " + x);
    }

    // generates an object that is immediately destroyed
    void generator(int i)
    {
        FDemo o = new FDemo(i);
    }
}
//-----
class Finalize
{
    public static void main(string args[])
    {
        int count;

        FDemo ob = new FDemo(0);

        /* Now, generate a large number of objects. At
        some point, garbage collection will occur.
        Note: you might need to increase the number of objects
        generated in order to force garbage collection. */

        for(count=1; count < 100000; count++)
            ob.generator(count);
    }
}
```

وبعد ترجمة البرنامج وعرضه يكون الخرج

```
Finalizing 58493
Finalizing 66085
Finalizing 66084
Finalizing 66083
Finalizing 66082
Finalizing 66081
Finalizing 66080
Finalizing 66079
Finalizing 66078
Finalizing 66077
```

```
.....
Finalizing 65974
Finalizing 65973
Finalizing 65972
Finalizing 65971
```

نسخة تجريبية - ليست للنشر

## مصفوفة كائنات Arrays & Collections

من الممكن إنشاء مصفوفة كائنات (مصفوفة موجهات لكائنات) يتم حجز عدد محدد من المواقع المتجاورة في الذاكرة لتخزين البيانات فيها ,  
إنشاء المصفوفة لا ينشئ كائنات انما تنشئ مجموعة من الموجهات. بناء مصفوفة الكائنات تتم بطريقتين: بناء مصفوفة الموجهات ثم بناء الكائنات .

```
class App
{
    public static void main(string[] args)
    {
        Point[] arr = new Point[5];

        for (int i=0; i<arr.length ; i++)
            arr[i] = new Point();

        arr[0].SetPoint(11,22);
        arr[1].SetPoint(33,44);
        arr[2].SetPoint(55,66);
        arr[3].SetPoint(77,88);
        arr[4].SetPoint(99,11);

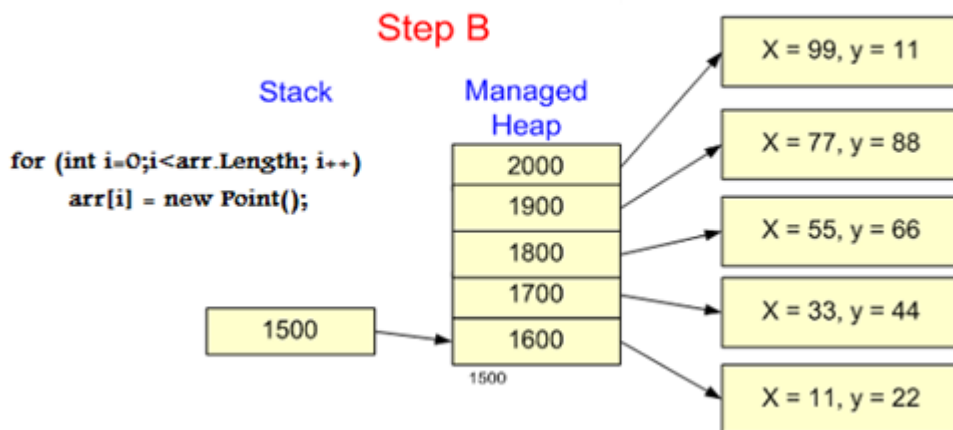
        for(Point o : arr)
        {
            o.Print();
        }

        for (int i=0; i<arr.length; i++)
        {
            arr[i].Print();
        }
    }
}
```

طريقة 1 للطباعة

طريقة 2 للطباعة

### وضعية الذاكرة تعرض بالصورة التالية



مثال – الدوال Add + Remove

```
public class Subject
{
    private string name;
    private int grade;

    public Subject(string name, int grade)
    {
        this.name = name;
        this.grade = grade;
    }
    public string GetName()
    {
        return this.name;
    }
    public void SetName(string name)
    {
        this.name = name;
    }
    public int GetGrade()
    {
        return this.grade;
    }
    public void SetGrade(int grade)
    {
        this.grade = grade;
    }
    public void PrintSubjectInfo()
    {
        System.out.println("Subject Name:"+this.name+" Grade :"+this.grade);
    }
}
```

```
class Student
{
    private string name;
    private subject AllSub[];

    public Student(string name, int NumSub)
    {
        this.name = name;
        this.AllSub = new subject[NumSub];
    }
    public string GetName()
    {
        return this.name;
    }
    public void SetName(string name)
    {
        this.name = name;
    }
}
```

```
public void AddSubject(subject sub1)
{
    int i;
    for (i = 0; i < AllSub.Length; i++)
        if (this.AllSub[i] == null)
        {
            this.AllSub[i] = sub1;
            break;
        }
}
public void RemoveSubject(subject sub1)
{
    int i;
    for (i = 0; i < AllSub.Length; i++)
        if (this.AllSub[i] != null)
        {
            _____
            _____
            _____
            _____
        }
}
public double GetAVG()
{
    int i, sum = 0, count = 0;
    double avg=0;

    for (i = 0; i < AllSub.Length; i++)
    {
        if (this.AllSub[i] != null)
            sum += this.AllSub[i].GetGrade();
        count++;
    }
    if (count > 0) avg = (double)sum / count;
    return avg;
}
}
```

```
class Program
{
    static void main(string[] args)
    {
        Student s1=new Student("hisham", 4);
        Subject sub1=new Subject ("Math" , 100);
        s1.AddSubject(sub1);
        s1.AddSubject(new Subject("Heb",70));
        s1.AddSubject(new Subject("Eng",80));
    }
}
```

## تمارين – كائنات ومصفوفة كائنات

### سؤال 1 :

معطاة الفئة التالية

إدعاء خروج	رأس العملية
بان يتلقى قيمة جديدة لـ X ثم ينشئ كائناً جديداً	A (int x)
الباني الناسخ	A (A a)
تعيد قيمة X	int GetX( )
تغير قيمة X	void SetX(int x)
تعيد قيمة X كنص وفق النسق التالي: x is: <X>	string toString( )

كتبنا بناء على الفئة A المقاطع التالية:

<pre>public class B {     private A a;     private int y;     public void SetA()     {         this.a.SetX(17);     }     public string ToString()     {         return a.ToString() + " y is: " + this.y;     } }</pre>	<pre>class Program {     static void main(string[] args)     {         A a1 = new A(6);         B b = new B(a1, 11);         A a2 = new A(a1);         b.SetA();         a1.SetX(36);         Console.WriteLine(a1.ToString());         Console.WriteLine(a2.ToString());         Console.WriteLine(b.ToString());     } }</pre>
--	--

أ) برمج الفئة A

ب) نريد كتابة الباني الناسخ في الفئة B. أي حل من بين الحلين التاليين صحيح وأيهم خطأ؟ بيّن لماذا؟ وحدد بالنسبة للحل الصحيح المخرج.

<pre>public B(A a, int y) {     this.a = a;     this.y = y; }</pre>	<pre>public B(A a, int y) {     this.a.SetX(a.GetX());     this.y = y; }</pre>
---	--



ج) تتبع عمل المقطع المعطى أعلاه وحدد المخرج مبينا وضع الكائنات والمؤشرات في كل خطوة.

د) أضفنا إلى الفئة A الخاصية

```
private int p;
```

وغيرنا في الفئة B العملية SetA كالتالي:

```
public void SetA()  
{  
    this.a.SetX(17);  
    this.a.p = 17;  
}
```

هل التغيير الذي أجريناه صحيح؟ علل الإجابة.

نسخة تجريبية - ليست للنشر

## سؤال 2:

أمامك uml للفتة :Song

Song
string name; int length;
Song(string s,int l) Song(Song s); string GetNameSong(); int GetLengthSong(); void SetName (string s); string toString();

أ. أكتب الدالة void SetName (string s)  
ب. أكتب ال- ctor وأيضا copy ctor إذا تواجد?  
أكتب بشكل واضح

أمامك uml للفتة :Disc

Disc
const int MAX_SONGS=10; string nameDisc; Song songs[ ]; Int currentSong;
Disc(string s); Disc (Disc d); string GetNameDisc(); Song[ ] GetAllSongs(); void AddSong(Song s); void SetNameDisc(string s); int GetCurrentSong(); string toString();

أكتب البنائين.  
أكتب الدالة string ToString( )  
أكتب الدالة GetAllSongs( )  
أكتب الدالة AddSong( )

- (أ)
- إنشاء كائن من نوع Song بمواصفات من اختياركم
  - إنشاء كائن آخر بمواصفات الكائن الأول عن طريق الباني الناسخ .
  - فحص هل الكائنات متساويان عن طريق الدالة Equals وطباعة جملة توضيحية
  - طباعة خصائص الكائنين (الاغنييتين)
  - إنشاء كائن من نوع Disc وأضف الاغنيتين الى الاسطوانة
  - هل العملية التالية صحيحة (أي لا تحدث أخطاء Compiler) ولماذا؟
- ما الحل اذا كانت خطأ؟

```
Song s1 = new Song();
```

ب ( هذا جزء من الدالة الرئيسية :main

```
int n=int.Parse(Console.ReadLine());
string s;
Disc d = new Disc("sky");

for ( i = 1; i < n; i++)
{
    s = Console.ReadLine();
    d.AddSong(new Song(s, int.Parse(Console.ReadLine())));
}
```

- أستقبل أسم أغنية موجودة على الاسطوانة , وأسم أغنية جديدة يبدل أسم الاغنية الموجودة على الاسطوانة باسم الاغنية الجديد
- أحسب معدل الزمن لجميع اطوال الاغاني على الاسطوانة (معدل زمن جميع الاغاني) .

### سؤال 3 :

الرزنامة عبارة عن مصفوفة من الأحداث. لكل حدث يوجد تاريخ بداية ووقت بداية ووصف.

- Calendar – فئة تمثل النوع رزنامة
- Event – فئة تمثل النوع حدث
- Date – فئة تمثل النوع تاريخ (معطاة ولا حاجة لبرمجتها)
- Time – فئة تمثل النوع وقت (معطاة ولا حاجة لبرمجتها)

#### الفئة Calendar

على الفئة أن تحتوي على الخصائص والدوال التالية:

#### الخصائص:

Year : السنة. خاصية من نوع int

Events : مصفوفة من الأحداث بحجم 365. خاصية من نوع Event

#### العمليات:

- بان : يأخذ كبارامتر السنة وينشئ المصفوفة.
- toString : تعيد تفاصيل الرزنامة وفق النسق التالي:

Year: 2012

Event1:

StartDate = 1/12/2012

StartTime = 12:00 am

Description: Meeting with the manager

Event2:

StartDate = 2/12/2012

StartTime = 11:00 am

Description: Staff Meeting

- دالة باسم Add تأخذ كبارامتر كائنا من نوع Event وتضيفه إلى مصفوفة الأحداث وتعيد true إن كان له مكان وإلا فتعيد false.
- دالة باسم Remove تأخذ كبارامتر كائنا من نوع Event وتحذفه من مصفوفة الأحداث وتعيد true إن تم حذفه بنجاح وإلا فتعيد false.
- دالة باسم PrintEvents تأخذ كبارامتر كائنا من نوع Date باسم StartDate وتطبع جميع الأحداث التي تبدأ في StartDate

## الفئة Event

لكل حدث يوجد: ( وصف , تاريخ البداية , ساعة البداية )

### على الفئة أن تحوي :

- بانياً يأخذ قيماً لجميع الخصائص
- دالة Equals
- دالة toString تعيد خصائص الحدث وفقاً للنسق التالي:

StartDate = 1/12/2012

StartTime = 12:00 am

Description: Meeting with the manager

- أرسم مخطط UML للفئات المذكورة أعلاه. على المخطط أن يحتوي على جميع خصائص الفئتين Event و Calendar. لا حاجة لكتابة خصائص الفئتين Date و Time في المخطط.
- عليكم برمجة الفئات Event و Calendar. افترض وجود Setters و Getters في الفئات.
- ما هي درجة تعقيد العملية PrintEvents؟ علل الإجابة؟
- أكتب برنامجاً يفحص الفئة Calendar وفقاً لما يلي:
  - ينشئ كائناً جديداً بالموصفات المطبوعة أعلاه (عند الدالة toString).
  - يطبع مواصفات هذا الكائن.
  - يطبع جميع الأحداث التي تبدأ بتاريخ 2011/12/12

## الفئتان Date و Time (معطاة ولا حاجة لبرمجتها)

```
public class Date
{
    private int Day, Month, Year;

    public Date(int Day, int Month, int Year)
    {
        this.Day = Day;
        this.Month = Month;
        this.Year = Year;
    }

    // Getters and Setters ...

    @Override public String toString()
    {
        return Day + "/" + Month + "/" + Year;
    }
}

public class Time
{
    private int Hour, Minuts;
    private String AmPm;

    public Time(int Hour, int Minuts, String AmPm)
    {
        this.Hour = Hour;
        this.Minuts = Minuts;
        this.AmPm = AmPm;
    }

    // Getters and Setters ...

    @Override public String toString()
    {
        return Hour + ":" + Minuts + " " + AmPm;
    }
}
```

## سؤال 4:

أكتب فئة باسم **TV** يمثل النوع "تلفاز". الصف يحتوي على الخصائص والدوال التالية:

- ماركة التلفاز (نص).
- قطر الشاشة (عدد).
- الثمن (عدد).
- بانياً يأخذ كباراً مترات قيمة لجميع الخصائص وينشئ تلفازاً وفقاً لهذه المعطيات.
- Copy Constructor.
- Get'er و Set'er لخاصية واحدة فقط.
- دالة باسم Equals تأخذ كباراً متر تلفازاً وتعيد "صدق" إذا كان مساوياً للكائن الحالي. خلاف ذلك تعيد "كذب".
- دالة ToString تعيد خصائص التلفاز كنص ووفقاً للنسق التالي:  
TV Data: (Orion ,60, 6000)

### الفئة TVStore

- تمثل هذه الفئة دكاناً لبيع أجهزة تلفاز. على الفئة أن تحوي
- عنوان الصف وتعريف خصائصه
  - بانياً افتراضياً ينشئ مصفوفة الأجهزة بحجم 30
  - بانياً يأخذ كباراً متر عدداً طبيعياً يمثل حجم مصفوفة الأجهزة ثم ينشئ المصفوفة بهذا الحجم.
  - دالة toString تعيد خصائص الأجهزة الموجودة في الدكان وفق النسق المشروح ألاه لكل جهاز.
  - دالة باسم Add تأخذ كباراً متر كائناً من نوع TV وتضيفه إلى مصفوفة الأجهزة وتعيد true ان كان له مكان والا فتعيد false.
  - دالة باسم Remove تأخذ كباراً متر كائناً من نوع TV وتحذفه من مصفوفة الأجهزة وتعيد true ان تم حذفه بنجاح والا فتعيد false.

G. دالة باسم Remove تأخذ كبرامتر رقم خلية من خلايا مصفوفة الأجهزة ثم تقوم بحذف الجهاز الموجود في هذه الخلية ان وجدت في الخلية تلفازا وتعيد true والا فتعيد false.

H. دالة باسم Find تأخذ كبرامترات مواصفات تلفاز يريده زبون ما (الماركة, قطر الشاشة, قيمة كبرى للثمن). الدالة تبحث عن جهاز مناسب وتعيده ككائن إن وجدته وإلا فتعيد null.

I. دالة باسم Find تأخذ كبرامتر كائنا من نوع TV بمواصفات التلفاز الذي نريد البحث عنه. الدالة تبحث عن جهاز مناسب وتعيده ككائن إن وجدته وإلا فتعيد null.

### الفئة TVStoreTester

أكتب برنامج (main) للصف TVStore بحيث تنشئ شركة عندها 3 أجهزة مختلفة, ومن ثم تستعمل الدالة Find مع جمل إيضاح مناسبة وطباعة خصائص التلفاز المناسب إن وجد .



## سؤال 5:

أ) أكتب صفًا باسم (QuadraticE) يمثل النوع "معادلة تربيعية". الصيغة العامة لها هي:

$$ax^2 + bx + c = 0$$

الفئة تحتوي على الخصائص والدوال التالية:

- خاصية من نوع int باسم a .
  - خاصية من نوع int باسم b .
  - خاصية من نوع int باسم c .
  - بانياً يأخذ كبارامترات قيماً لجميع الخصائص وينشئ معادلةً جديدةً .
  - Get'er و Set'er لخاصية واحدة فقط (من اختياركم) .
  - دالة باسم GetNumOfSolutions تعيد عدد حلول المعادلة.
  - دالة باسم GetSolutions تعيد حلول المعادلة كنص في النسق التالي
- $$x1= \dots x2= \dots$$
- دالة باسم Equals تأخذ كبارامتر معادلةً تربيعيةً وتعيد "صدق" إذا كانت مساويةً للكائن الحالي. خلاف ذلك تعيد "كذب".
  - دالة toString تعيد خصائص المعادلة كنصٍ ووفقاً للنسق التالي :
- $$2x^2 + 3x - 2 = 0$$

على افتراض أن  $a = 2$   $b = 3$   $c = -2$

ب) استعمل الصف QuadraticE بالصف QuadraticETester والذي يحتوي على دالة main.

عليكم تنفيذ التالي:

- إنشاء كائنين بمواصفات من اختياركم .
- استعمال جميع الدوال كما تشاءون .

## سؤال 6 :

تريد شركة تأجير سيارات أن تحو سب عملية التأجير. لكل سيارة يوجد:

- اسم: وهو عبارة عن نص
- رقم: وهو عبارة عن نص
- سنة الإنتاج
- عدد الركاب
- السعر لليوم الواحد

من أجل ذلك الغرض مطلوب منك أن تكتب صفًا جديدًا باسم Car بالخصائص المعطاة أعلاه.

على الصف أن يحتوي أيضاً على العمليات التالية:

- بانٍ يأخذ كباراً مترات قيمةً لجميع الخصائص وينشئ سيارةً وفقاً لهذه المعطيات
- Copy Constructor
- Get'er و Set'er لخاصية واحدة فقط
- عملية باسم Equals تأخذ كباراً متر سيارةً وتعيد "صدق" إذا كانت مساوية للكائن الحالي. خلاف ذلك تعيد "كذب".
- عملية باسم toString تعيد خصائص السيارة كنصٍّ ووفقاً للنسق التالي:

Name = Opel Astra

Nr = 0123456789

Year = 2000

Passengers = 5

Price = 100

طبق المهام التالية :

- استعمل الصف Car بالصف CarTester والذي يحتوي على دالة main.
- إنشاء 5 سيارات أجرة بواسطة مصفوفة أحادية بمواصفات من اختياركم .
  - معروف لديك أن هنالك 2 سيارات بنفس التفاصيل فقط الأختلاف برقم السيارة التسلسلي (استعمل البناء الناسخ) .
  - دالة تجد عدد الركاب الكلي لجميع السيارات .
  - دالة تتلقى عدد الركاب (يمكن أن تتسع لهم سيارة واحدة ويمكن أن لا تتسع لهم سيارة واحدة) تعيد السعر لليوم الواحد . إذا كان عدد الركاب أكبر من جميع المقاعد في جميع السيارات تعيد 1- .
  - استعمال جميع العمليات كما تشاءون .

مكتبة  
البرمجة  
الهندسية  
للنشر

## سؤال 7 :

أ) أكتب صفًا جديدًا بإسم Book يمثل النوع كتاب. على الصف أن يحتوي على الخصائص والعمليات التالية:

### الخصائص:

Title : اسم الكتاب. خاصية من نوع string  
ISBN : الرقم العالمي للكتاب. خاصية من نوع int  
Author : اسم مؤلف الكتاب. خاصية من نوع string

### العمليات:

- Constructor : يأخذ قيمًا لجميع الخصائص وينشئ كائنًا جديدًا.
- Copy Constructor
- toString : تعيد التفاصيل عن الكتاب وفق النسق التالي:

Title: Programming in Java

ISBN: 12345

Author: Sun Microsystems

- SetISBN : حدد ما الذي يجب أن تأخذه هذه العملية وأكتبها
- HasISBN : تأخذ كبراً مراً عدداً صحيحاً وتعيد true إذا كان العدد مساوياً لرقم الكتاب الحالي.

ب) أكتب صفًا باسم BookTester يفحص الصف Book وفقاً لما يلي:

- ينشئ كائنًا جديدًا بالموصفات المطبوعة أعلاه (عند الدالة ToString).
- يطبع مواصفات هذا الكائن.
- يغير الرقم العالمي للكتاب.
- يطبع مواصفات هذا الكائن مرةً أخرى.

## سؤال 8:

رسالة بريد الكتروني EMessage تكون مكونة من عدة أمور:

- قائمة عناوين البريد الإلكتروني للمرسل إليهم .
  - موضوع الرسالة
  - نص الرسالة
  - مجموعة ملفات مرفقة مع الرسالة. الذي يهمنا هنا فقط أسماء الملفات. عدد الملفات لا يجوز أن يزيد عن 10 ملفات.
1. أكتب الصف EMessage بلغة C#. على الصف أن يشمل على:

1.1. عنوان الصف وتعريف خصائصه.

1.2. دالة ToString لإعادة خصائص الرسالة وفق النسق التالي:

SendTo: a1@b.com, a2@c.com,...

Subject: <...>

Body: <...>

Attachments: File1, File2, ...

البريد الإلكتروني لشخص ما يكون مكوناً من عدة أمور:  
عنوان بريد الشخص , قائمة الرسائل الجديدة, قائمة الرسائل المرسله, قائمة الرسائل المحذوفة.

- أكتب الصف EMail بلغة C#. على الصف أن يشمل على:
  - عنوان الصف وتعريف خصائصه.
  - دالة باسم Remove تأخذ رسالةً كبرامتر وتحذفها. عملية الحذف هذه تبحث عن الرسالة في قائمة الرسائل الجديدة وقائمة الرسائل المرسله بناءً على اسم الرسالة فإذا وجدت الرسالة تنقلها إلى قائمة الرسائل المحذوفة.
  - دالة باسم RemoveAllDeleted تحذف جميع رسائل قائمة الرسائل المحذوفة. على الدالة أن تسأل المستخدم إذا كان حقاً يريد تنفيذ عملية الحذف هذه.

- ما هو نوع العلاقة بين الفئة EMessage والفئة EMail. أرسم مخطط UML يوضح هذه العلاقة.
- أكتب دالة باسم Search تأخذ نصاً معيناً كبرامتر وتطبع جميع الرسائل التي يظهر النص على الأقل في واحدةٍ من خصائصها.
- أين يجب إضافة الدالة Search? ولماذا?

## سؤال 9 :

أ) أكتب صفًا جديدًا بإسم FootballTeam يمثل النوع "فريق كرة قدم".

### الخصائص:

Name : اسم الفريق. خاصية من نوع string

Trainer : اسم المدرب. خاصية من نوع string

Wins : عدد الانتصارات. خاصية من نوع int. قيمتها الأولية 0.

Losses : عدد الخسارات. خاصية من نوع int. قيمتها الأولية 0.

### العمليات:

• بانياً : يأخذ قيماً لجميع الخصائص ما عدا Wins و Losses.

• ToString : تعيد تفاصيل الفريق وفق النسق التالي:

Name: Abnaa Sakhnin

Trainer: No Name

Wins: 4 Losses: 6

• Win : ترفع عدد الانتصارات بواحد

• Loose : ترفع عدد الخسارات بواحد.

• Percentage : تعيد نسبة الانتصارات من جميع الألعاب: عدد الألعاب / عدد الانتصارات.

• BetterThan : تأخذ كبارامتر كائناً من نوع FootballTeam وتعيد true إذا

كان فريق الكائن الحالي أفضل من فريق الكائن الممرر بواسطة البارامتر. الفريق

الأفضل هو الفريق الذي قيمة Percentage لديه أكبر. على هذه الدالة أن

تستعمل الدالة Percentage.

ب) أكتب صفاً باسم `Tester` يفحص الصف `FootballTeam` وفقاً لما يلي:

- ينشئ كائناً جديداً بالمواصفات المطبوعة أعلاه (عند الدالة `Print`).
- يطبع مواصفات هذا الكائن.
- يبلغ الكائن بأن الفريق إنتصر 3 مرات وخسر 8 مرات.
- ينشئ كائناً جديداً آخرًا بخصائص من اختياركم.
- يستعمل الدالة `BetterThan` لمعرفة أي الفريقين أفضل ويطبع جملة توضح ذلك.

نسخة تجريبية - ليست للنشر

# الوراثة

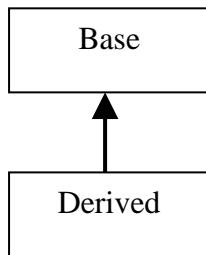
# Inheritance

نسخة تجريبية  
يجوز النشر



# الوراثة – Inheritance

الوراثة هي المبدأ الثاني من مبادئ برمجة الكائنات الموجهة Opp والتي يمكن الاستفادة منها في لغة جافا لتطوير البرامج حيث يمكن استخدام الفئات التي تم تصميمها وتنفيذها وتأكيدنا من أنها تعمل بصورة جيدة ثم نكتب فئة جديدة يضاف إليها الطرق والبيانات الجديدة فقط وترث الطرق الموجودة في الفئة القديمة التي يمكن اعتبار الفئة الجديدة امتداداً لها . الفئة الوراثة تسمى Derived والموروثة تسمى Base .



مثال : نفترض أننا نريد إنشاء حساب بنكي يضيف عائداً شهرياً على الرصيد الموجود في البنك.

بدراسة هذا النوع من الحساب أضح لنا أن هذا الحساب هو امتداد لحساب البنك التي تم برمجته في الفئة BankAccount حيث إنه يتطلب عمليات ايداع وسحب واستعلام عن رصيد وله متغير لتخزين نسخة من قيمة الحساب ولذلك يمكن استخدام مبدأ الوراثة لتطوير البرنامج السابق بإنشاء فئة جديدة ويضاف إليها دالة لحساب العائد ويضاف إليها أيضاً متغير لمعدل العائد الشهري. أنظر المثال التالي.

```

public class SavingAccount extends BankAccount
{
    private double interestRate;

    public SavingAccount(double rate)
    {
        interestRate = rate;
    }
  
```

```
// addInterest method
public void addInterest()
{
    double interest = getBalance()*interestRate/100;
    deposit(interest);
}

}
```

وبتحليل هذا المثال يمكن أن نستعرض بعض خصائص الوراثة في لغة جافا ومنها :

1. من أسباب استخدام الوراثة هو إعادة استخدام شفرة البرنامج `code reuse` حيث استخدام فئات موجودة ونوفر الجهد المبذول لإتقان تصميم وتنفيذ هذه الفصائل.

2. الفئات التي تورث تسمى الفئات العامة (العليا `superclass`) لأنها تحتوي على الطرق والبيانات المشتركة وأحيانا تسمى فئة الأب (`parent class`) أو الفئة الأساسية (`Base class`).

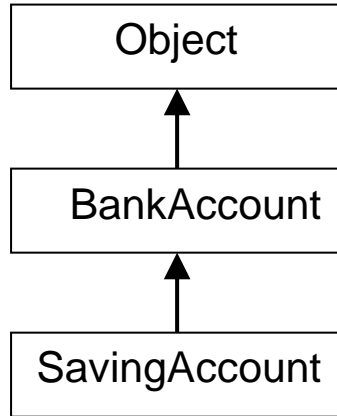
3. الفئات التي ترث تسمى الفئات الفرعية (`subclass`) لأنها تحتوي على الطرق والبيانات الخاصة المضافة وأحيانا تسمى فئات الابن (`child class`) أو الفئات المشتقة (`drived class`).

4. ولتحقيق عملية الوراثة وتوريث فئة قديمة إلى فئة جديدة عند إنشائها نقوم بكتابة اسم الفئة الجديدة ثم الكلمة المحجوزة الدالة على الوراثة `extends` التي تعني أن هذه الفئة هي امتداد للفئة القديمة التي يكون اسمها بعدها وتكون الصيغة العامة للوراثة هي :

`class subclass Name extends superclass Name`

5. عندما نقوم بإنشاء فئة ولم نحدد اسم فئة ترث منها تفترض لغة جافا أنك ترث من الفئة الأم `object` ومثال ذلك الفئة `BankAccount` ترث الفئة `object`.

6. الفئة object تحتوي على عدد صغير من الطرق التي تعني شيئاً لجميع الكائنات مثل الطريقة toString التي يمكن استخدامها للحصول على وصف حالة الكائن.



الرسمه تبين علاقة الوراثة بين الفصائل الثلاثة , SavingAccount , BankAccount, Object ويسمى مخطط الوراثة Inheritance Diagram.

عند إنشاء كائن جديد من الفئة الفرعية ينادي أولاً المنشئ، الموجود في الفئة العامة ليعطي قيمة مبدئية للمتغيرات الموجودة فيها ثم ينفذ المنشئ، الموجود في الفئة الفرعية لإعطاء قيم أولية للمتغيرات الجديدة.

## كلمة المفتاح Super:

كلمة المفتاح Super تسمح للفئة الوارثة بالوصول للخصائص والعمليات التي ورثتها. يُمكن إستدعاء بواني الفئات الموروثة. الإستدعاء يجب أن يكون داخل الباني للفئة الوارثة.

```
public Employee(String frst,String lst,int ag,int id,int nm,String dpt)
{
    super(frst,lst,ag,id);
    this.emp_num = nm;
    this.department = dpt;
}
```

الإستدعاء للباني الذي يتلقى 4 قيم

كلمة المفتاح Super تسمح إستدعاء الدوال والطرق ومنع حدوث تراجعية .  
مثال : الدالة toString يمكن أن تستدعي الدالة toString في الفئة الموروثة وزيادة بعض القيم.

```
@Override public String toString()
{
    String st = super.toString();
    st = st + "Emp Num:"+this.emp_num+",department:"+ this.department;
    return st;
}
```

إستدعاء للدالة toString الموجودة في الفئة الموروثة . ثم نضيف بعض القيم.

## الطرق-الدوال :Methods

عند تعريف طرق فئة فرعية يمكن أن يكون هناك ثلاثة احتمالات:

(1) استخدام الطريقة في الفئة الفرعية بنفس الاسم ونفس المعاملات (نفس البصمة/العنوان) كما في الفئة العامة ولكن هناك استبدال للجمل التنفيذية لهذه الطريقة وتسمى (override method) ولذلك عند استدعاء هذه الطريقة باستخدام متغير من نوع الفئة الفرعية فيتم تنفيذ الطريقة الموجودة في الفئة الفرعية وليست الطريقة الموجودة في الفئة العليا.

(2) يمكن للفئة الفرعية أن ترث الدوال الموجودة في الفئة العليا بدون أي تغيير, في هذه الحالة عند استدعاء هذه الطرق باستخدام كائن من نوع الفئة الفرعية يتم تنفيذ الطريقة من الفئة العليا.

(3) يمكن للفئة الفرعية إنشاء طرق جديدة لتحقيق الغرض من التطور وتستدعي هذه الطرق أو الدوال الجديدة فقط باستخدام كائنات الفئة الفرعية.

### متغيرات الكائنات (Instance variables)

بالنسبة لمتغيرات الكائنات يوجد حالتان فقط

جميع متغيرات الكائنات للفئة الفرعية تورث تلقائياً لكائنات الفئة الفرعية.

يمكن تعريف متغير جديد فقط على الكائنات الفئة الفرعية.

## وراثة بعدة أدوار (طبقات) (Chain Inheritance)

فئة وراثتها يمكن أن تكون هي أيضا موروثة (فئة أساسية) من قبل فئة أخرى .

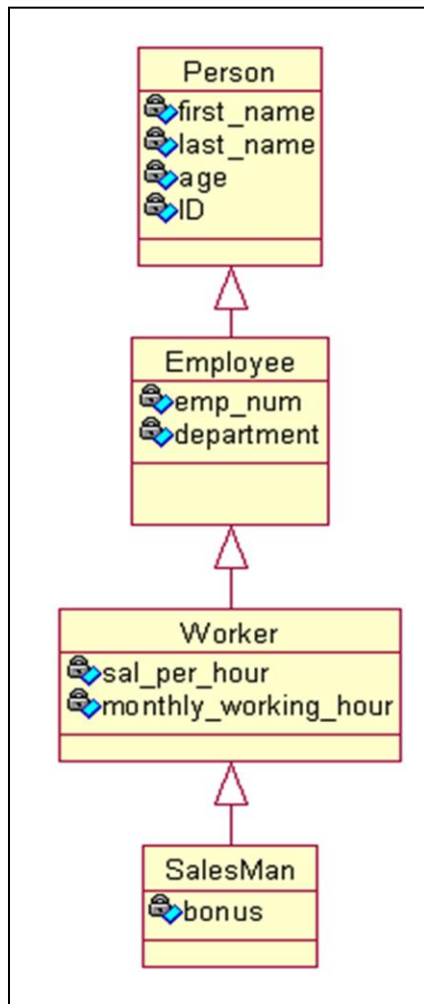
### مثال :

الفئة Person - هي فئة موروثة (Base) .  
كل الفئات الأخرى ترثها .

الفئة Employee - ترث الفئة الأساسية (Person)  
يعني ترث كل الميزات وكل الدوال والإجراءات لها .

الفئة Worker - ترث الفئة (Employee) التي هي  
أيضا ترث الفئة (Person) , يعني أن الفئة  
Worker ترث كل الميزات لـ Employee  
وأيضاً Person .

الفئة SalesMan - ترث الفئة (Worker) التي هي  
أيضا ترث الفئة (Employee) , يعني أن الفئة  
SalesMan ترث كل الميزات لـ Worker  
وأيضاً Employee وأيضاً Person .



```

class Person
{
    private String first_name;
    private String last_name;
    private int age;
    private int id;
    //-----
    public Person ()
    {
    }
    public Person(String first,String last,int age, int id)
    {
        this.first_name = first;
        this.last_name = last;
        this.age = age;
        this.id = id;
    }
}
  
```

```
public void SetPerson(String first,String last,byte age, int id)
{
    this.first_name = first;
    this.last_name = last;
    this.age = age;
    this.id = id;
}
public void PrintPerson()
{
    System.out.println("Name : " + last_name + " " + first_name);
    System.out.println ("id :" + id);
    System.out.println ("age:" + age);
}
}

-----

class Employee extends Person
{
    private int emp_num;
    private String department;

    public Employee ()
    {
    }
    public Employee (String first,String last,int age,int id,int
num,String dpt)
    {
        super(first,last,age,id);
        this.emp_num = num;
        this.department = dpt;
    }
    public void SetEmployee(String first,String last,byte age, int id,
int num , String dpt)
    {
        SetPerson(first,last,age,id);
        this.emp_num = num;
        this.department = dpt;
    }

    public void PrintEmployee()
    {
        PrintPerson();
        System.out.println("Emp num:"+ emp_num + "\nDep"+ department);
    }
}

-----
```

```
class Worker extends Employee
{
    private float sal_per_hour;
    private float working_hour;

    public Worker(String first, String last, int age, int id, int num ,
String dpt, float sal , float hours)
    {
        super(first,last,age,id,num,dpt);
        this.sal_per_hour = sal;
        this.working_hour = hours;
    }
    public void SetWorkerSalary(float s, float w)
    {
        sal_per_hour = s;
        working_hour = w;
    }
    public float CalcWorkerSalary()
    {
        return sal_per_hour * working_hour;
    }

    public void SetWorker(String ln, String fn ,byte age ,int id, int
num, String dpt,float sal,float hours)
    {
        SetEmployee(fn, ln, age, id, num, dpt);
        this.sal_per_hour = sal;
        this.working_hour = hours;
    }
    public void PrintWorker()
    {
        PrintEmployee();
        System.out.println("monthly salary = " + CalcWorkerSalary());
    }
}

-----

class SalesMan extends Worker
{
    private float bonus;

    public SalesMan(String first,String last,int age, int id, int num
        ,String dpt, float sal , float hours , float bonus)
    {
        super(first,last, age,id,num,dpt,sal,hours);
        this.bonus = bonus;
    }
    public void SetSalesMan (String first,String last,byte age,int id, int
        num ,String dpt,float sal ,float hours ,float  bonus)
    {
        SetWorker(first,last,age,id,num,dpt,sal,hours);
        this.bonus = bonus;
    }
}
```



```
public void SetSalesManSalary(float sal,float hours , float bonus)
{
    SetWorkerSalary(sal,hours);
    this.bonus = bonus;
}
public float CalcSlaesManSalary()
{
    return CalcWorkerSalary() + bonus;
}
public void PrintSalesMan()
{
    PrintEmployee();
    System.out.println("monthly salary : " + CalcSlaesManSalary());
}
}
```

```
-----
class App
{
    static void main(String[] args)
    {
        SalesMan s = new
        SalesMan("Yossef","Zhrani",23,123,123,"Dpt1",34.4f,213,200);

        Worker w = new Worker("Safi","Salami",35, 4567, 4567,"Dpt2",
        37.45f, 198);

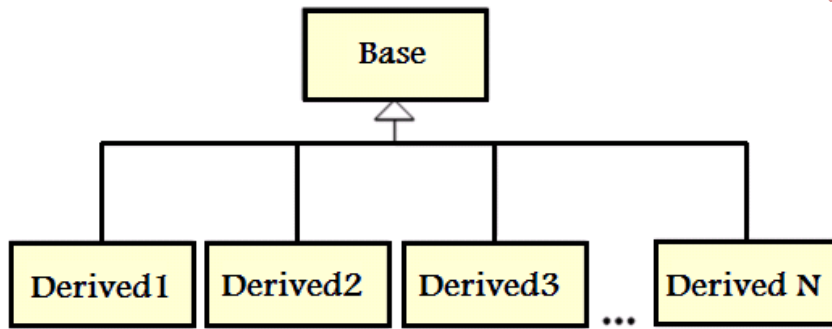
        s.PrintSalesMan();
        System.out.println("*****");

        w.PrintWorker();
        System.out.println("*****");
    }
}
```

## المخطط الهرمي للوراثة (Inheritance Hierarchies)

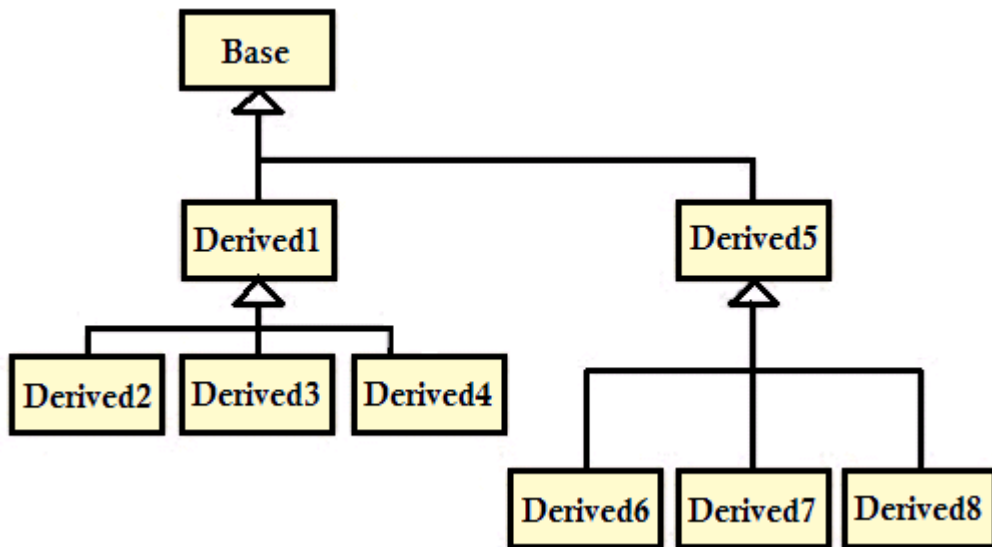
المخطط الهرمي عادة يمثل كشجرة , حيث معظم الفصائل التي تمثل المفاهيم العامة قريبة من الجذر (root) والفئات الاخرى الأكثر تخصصية نحو الفروع (branches) وسوف نستخدم مثال بسيط للمخطط الهرمي لإستكمال دراسة مفاهيم الوراثة .

فئة معينة يمكن أن تكون أساس لعدة فئات أخرى :

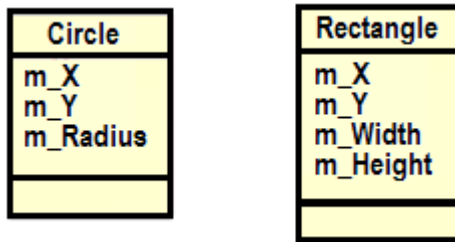


كل واحدة من الفئات الوارثة يمكن أن تكون هي أيضا موروثة (فئة أساسية) من قبل فئات أخرى .

مثال :



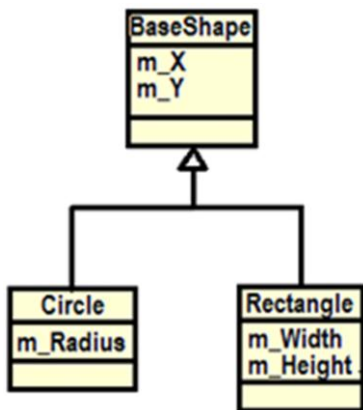
يمكن أن نصف الفئات التي تمثل دائرة ومستطيل بالشكل التالي :



المشكلة هي بتكرار الميزات وتكرار قسم من الدوال الداخلية , هذه المشاكل يمكن ان نحلها بواسطة عملية الوراثة . نجد الاشياء المشتركة بين الفئات نخرجهم من الفئات

ونبني منهم فئة أساسية (baseShape)

نصف مخطط الوراثة على هذا النحو :



### مثال :

طبق الفئات التي ذكرت في المثال السابق وأدخل الدالة الرئيسية التالية , ثم أكتب النتيجة.

```

class App
{
    static void Main()
    {
        Rectangle r = new Rectangle(10,20,120,200);
        Circle c = new Circle(300,350 ,75);
        r.PrintRectangle();

        System.out.println("Rect area= {0}",r.CalcArea());
        System.out.println("Rect perimeter =" +r.CalcPerimeter());

        c.PrintCircle();

        System.out.println ("Circle area = {0}",c.CalcArea());
        System.out.println ("Circle perimeter =" + c.CalcPerimeter());
    }
}
  
```

## Protected Modifier - متغير من نوع Protected

طرق ودوال للفئات الوارثة (Derived Class) لا يمكنها إستدعاء للدوال الخاصة للفئات الأساسية (Base Class). أحياناً نريد للفئات الوارثة تصل وتفعّل دوال موجودة في الفئات الأساسية , نريد أن نحسن طرق التنفيذ للبرمجيات . نريد أن نصل للدوال والخصائص في الدوال الموروثة بصورة مباشرة في الفئات الموروثة (الأساسية) أو لأجزاء منها .

لغة الجافا مثل كل لغات برمجة الكائنات الموجهة تعرّف طريقة الوصول ( Access Modifier). الميزات أو الدوال التي تعرّف كمتغير محمي - protected لديهم حماية معيّنة . لا يكمن الوصول اليه إلا من خلال الفئة التي عرّفت منه أو الفئات المشتقة (الوارثة) من هذه الفئة. **protected** تمكّن الفئات المشتقة وصول مباشر الى أعضاء (ميزات – دوال) في الفئة الأساسية دون الحاجة لأي تغيير , بكلمات أخرى هذه الميزات أو الدوال عامة للفئة الأساسية والفئات الوارثة.

```
class Base
{
    private int m_Num1;
    protected int m_Num2;
    public int m_Num3;

    public Base()
    {
    }
    public Base(int n1,int n2,int n3)
    {
        m_Num1 = n1;
        m_Num2 = n2;
        m_Num3 = n3;
    }
    protected void PrintBase()
    {
        System.out.println("Num1 = " + m_Num1);
        System.out.println("Num2 = " + m_Num2);
        System.out.println("Num3 = " + m_Num3);
    }
}
```

```
public int getNum1()
{
    return this.m_Num1;
}
public void setNum1(int n1)
{
    this.m_Num1 = n1;
}

public int getNum2()
{
    return this.m_Num2;
}
public void setNum2(int n1)
{
    this.m_Num1 = n1;
}
}

//-----
class Derived extends Base
{
    private String m_Str = "Derived class";

    public Derived()
    {
    }
    public Derived(int n1,int n2,int n3, String str)
    {
        super(n1,n2,n3);
        this.m_Str = str;
    }
    public String getStr()
    {
        return this.m_Str;
    }
    public void setStr(String s1)
    {
        this.m_Str = s1;
    }

    public void PrintDerived()
    {
        super.PrintBase();
        System.out.println("str = " + m_Str);
    }
    public void SetDataMembers(int n1,int n2,int n3, String str)
    {
        setNum1(n1);
        m_Num2 = n2;
        m_Num3 = n3;
        m_Str = str;
    }
}
```

```

class App
{
    static void main()
    {
        Derived d = new Derived(11, 22, 33 , "Moner");
        d.PrintDerived();

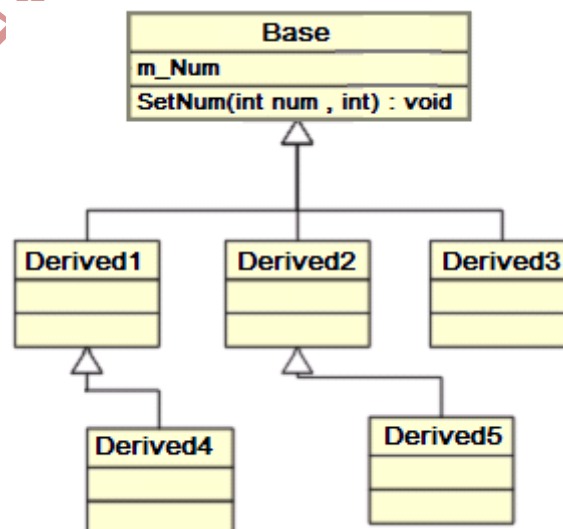
        d.SetDataMembers(10,20 ,30 , "Zhahi");
        d.PrintDerived();

        d.setNum1(55);
        d.setNum2(66);
        d.m_Num3 = 77;
        d.setStr("Zezo");
        d.PrintDerived();
    }
}

```

إستعمال متغير من نوع **protected** يجب أن يكون بصورة حذرة. لأنه يمكن الوصول للمتغيرات أو الميزات هذه بصورة مباشرة ولا حاجة لدالة عامة تصل للميزة. يعني أننا نستطيع الوصول بدون الدوال (get'ers , set'ers) . هذا الأمر يمكن أن يؤدي الى أخطاء منطقية يمكن أن يكون صعب كشفها وتصحيحها.

ننظر إلى المبنى الهرمي التالي كمثال :



نعتبر أن الميزة `m_Num` المعرفة داخل الفئة `Base` معرف من نوع `protected` ومجال قيمه من 0 حتى 100 .  
الدالة `super.SetNum()` تتلقى قيمة وتفحص قانونية القيمة المتلقاه قبل تعويضها بالميزة .

لكن هل الشئ ممكن أن يكون صحيح إذا عرفت الميزة من نوع `protected` ؟  
الجواب : لا

كل دالة – إجراء (طريقة) موجودة داخل الفئة ( `Derived1` حتى `Derived8` )  
يمكنها الوصول الى هذه الميزات بصورة مباشرة بدون الحاجة لإستعمال الدالة `super.SetNum()` , وهذا يمكنه أن يؤدي الى إحتمالات تعويض قيمة غير صحيحة داخل هذه الميزة .

## Overriding and Hiding Methods (إخفاء وإعادة كتابة)

من الممكن أن أرث طريقة معينة موجودة داخل الفئة الموروثة (Base) ولكن الفئة الوارثة (Derived) تريد كتابة هذه الطريقة حسب حاجتها , لغة الجافا بها مبدأ ما يسمى بإعادة كتابة العمليات Methods Overriding للفئة الوارثة .

مثال الطريقة أو العملية toString الموجودة داخل الفئة object وهي عبارة عن فئة موجودة في لغة الجافا وتجعلها بشكل تلقائي ال super class لجميع الفئات التي تكتب بلغة الجافا.

لغة الجافا بها مبدأ ما يسمى إخفاء دالة أو عملية Hiding Methods في الفئة الوارثة. بمعنى إذا كان لدينا دالة أو عملية داخل الفئة الموروثة لها عنوان مشابه لدالة موجودة داخل الفئة الوارثة يكون لدينا عملة إخفاء للدالة الموجودة داخل الفئة الموروثة عن طريق الدالة الموجودة داخل الوارثة.

If a subclass defines a class method with the same signature as a class method in the superclass, the method in the subclass hides the one in the superclass.

مثال على إخفاء دالة .

```
public class Animal
{
    public static void testClassMethod()
    {
        System.out.println("The class" + " method in Animal.");
    }
    public void testInstanceMethod()
    {
        System.out.println("The instance " + " method in Animal.");
    }
}
```



**The second class, a subclass of Animal, is called Cat:**

```

public class Cat extends Animal
{
    public static void testClassMethod()
    {
        System.out.println("The class method" + " in Cat.");
    }
    public void testInstanceMethod()
    {
        System.out.println("The instance method" + " in Cat.");
    }

    public static void main(String[] args)
    {
        Cat myCat = new Cat();
        Animal myAnimal = myCat;
        Animal.testClassMethod();
        myAnimal.testInstanceMethod();
    }
}

```

الفئة Cat ترث الفئة Animal وتعيد صياغة الدوال أو الطرق لها حسب حاجتها .  
 الفئة Cat تخفي الدوال التي لها نفس العنوان في الفئة الموروثة , يوجد عملية في الفئة  
 الوارثة بإسم testClassMethod() وعملية testInstanceMethod() .

**المخرج للبرنامج :**

The class method in Animal.  
 The instance method in Cat.

**تلخيص :**

الجدول التالي يلخص ماذا يحدث عندما نعرّف طرق أو دوال مع نفس العنوان مشابه  
 في الفئة الوارثة والموروثة .

طريقة static داخل الفئة الوارثة	طريقة داخل الفئة الوارثة	
Generates a compile-time error	Overrides	طريقة داخل الفئة الموروثة
Hides	Generates a compile- time error	طريقة static داخل الفئة الموروثة

## تمارين في الوراثة

### سؤال :

- أ) أية آلية برمجة كائنات موجهة تنعكس في تعريف بوان عدة في نفس الفئة ؟ كيف يقرر المترجم أي عملية بنائية ينفذ ؟
- ب) أمامك مقطع برنامج من الفئة A , في الفئة قسم من العمليات البنائية صحيحة وقسم خاطئ. بيّن أي منها صحيح وأي منها خاطئ ووضح سبب الخطأ .

```
public class A
{
    private int x;

    public A (int a)
    { this.x = a; }

    public A()
    { return new A(5); }

    public A (double b)
    {
        Super(b);
    }
}
```

- ج) افرض أن الكائن a1 بُني بصورة صحيحة : `A a1 = new A();`  
عرّف ثم طبق في الفئة A دالة بنائية التي تمكن بناء كائن بالصورة التالية:
- `A a2 = new A (a1);`

- د) أمامك فئتين :

```
public class C
{
    public void func1()
    {
        System.out.println("C: func1");
        func2();
    }

    public void func2()
    {
        System.out.println("C: func2");
    }
}
```

```
public class D extends C
{
    public void func2()
    {
        System.out.println ("D: func2");
    }
}
```

ماذا يطبع البرنامج التالي - المخرج :

```
public class Test
{
    public static void main (String[ ] args)
    {
        C c1 = new C( );
        D d = new D( );
        c1.func1( );
        d.func1( );
        d.func2( );
    }
}
```

### الحل :

السؤال يفحص الاشياء التالية **overloading**, استعمال العمليات البنائية , وراثه , وتعريف دوال وعمليات جديدة بالوراثة .

**حل أ:** مبدأ التحميل الزائد هذه الميزة تمكن إستعمال دوال بنفس الاسم داخل الفئة . اللغة تفرق بين الدوال ذات نفس الاسم بواسطة عدد البارامترات المستقبلية أو بواسطة نوع البارامترات , لا يمكن التفرق بين الدوال حسب القيمة المعادة او اسم البارامترات المرسله

### حل ب:

ثلاث عمليات بنائية الاولى صحيحة , الثانية غير صحيحة تستعمل بداخلها الامر **return** وهذا غير ممكن , الثالثة غير صحيحة لأنها تتوجه لعملية بنائية في الفئة الاساسية مع انها غير وراثه .

### حل ج:

```
public A (A a)
{
    this.x = a.x;
}
```

### حل د:

**C: func1**

C: func2

C: func1

C: func2

C: func1

C: func2

# تعدد الأشكال -

# Polymorphism

نسخة تجريبية - ليست للنشر

# تعدد الأشكال – Polymorphism

تعدد الأشكال هي المبدأ الثالث من مبادئ برمجة الكائنات الموجهة Opp والتي يمكن الاستفادة منها في لغة جافا لتطوير البرامج . حتى نستطيع تطبيق مبدأ تعدد الأشكال يجب أن يكون لدينا عملية وراثية .

الوراثة هي علاقة "شكل من أشكال ...." أو باللغة الإنجليزية Is A .  
أمثلة على ذلك :

السيارة هي شكل من أشكال وسائل النقل – (A Bus is A Vehicle)  
الطاولة هي شكل من أشكال الأثاث – (A Table is A furniture)

## مثال – برنامج لإدارة مدرسة:

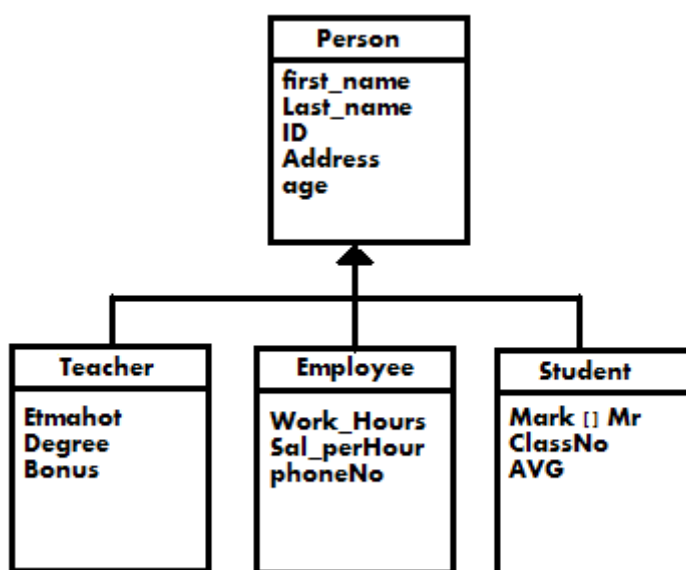
حتى نفهم هذا المبدأ بصورة أفضل , نفرض أننا نريد بناء برنامج يعالج أمور المدرسة (المدرسة تتألف من الفئات التالية : عامل, طالب , نائب مدير, معلم, مدير , ... ) .  
طبعاً كل فئة والميزات الخاصة بها .

نريد أن نحسب لكل طالب المعدل العام , لكل معلم أن نحسب الأجرة الشهرية له,  
للعامل نريد أن نعرف ساعة بداية العمل وساعة النهاية, للمدير نريد أن نضيف له زيادات معينة على الأجرة الشهرية .

نفرض أن لدينا في المدرسة 800 كائن من عدة فئات (مثلاً 700 طالب , 70 معلم الخ)

**سؤال :** كيف نبني البرنامج لإدارة المدرسة إذا فرضنا أننا نريد إستعمال مصفوفة الكائنات, من أي نوع نبنيها ؟ ربما نحتاج الى مصفوفات من الكائنات المختلفة وليس واحدة, كم نحتاج ؟

**الجواب :** بواسطة مبدأ تعدد الأشكال نستطيع أن نبني مصفوفة كائنات واحدة كبيرة من نوع الفئة الأساسية (Person) . وكل خلية من خلايا المصفوفة (التي هي بالأساس موجّهات) توجه الى كل كائن أياً كان من الفئات المشتقة .



إذا كانت الثلاثة فئات ترث الفئة الأساسية (Person) إذا مُوَّجه - مؤشّر reference من الفئة person يمكنه أن يُوجّه-يؤشّر على كائنات من نفس نوعه أو كائنات من الفئات المشتقة .

يمكن لهذا المُوجّه-المؤشّر أن يُفَعِّل دوال- طرق-إجراءات من الفئة الأساسية ولكنه لا يمكن أن يُفَعِّل دوال-طرق-إجراءات من الفئات المشتقة .

```
public static void main ( )
{
    Person p1 = new Student (.....);
    Person p2 = new Employee(....);
    Person p3 = new Teacher(....);
}
```

المُوجّه - المؤشر P1 يمكن أن يُفعّل دوال-

طرق-إجراءات من الفئة person

لكن لا يمكنه أن يُفعّل دوال-طرق-إجراءات

من الفئات Teacher , Employee , Student

الحالة نفسها للموجهات P2 , P3 .

صحيح أن الموجه- المؤشر P2 يُوّشر على كائن من نوع الفئة Employee لكن لا

يسمح له بإستعمال أو إستدعاء دوال - طرق داخل هذه الفئة المشتقة .

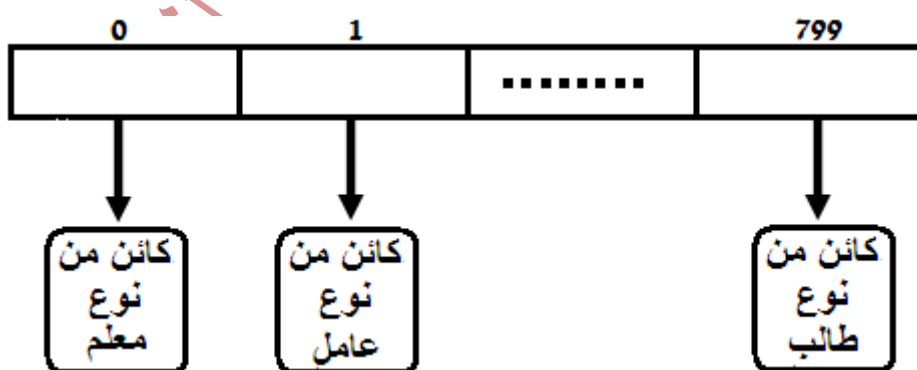
المؤشر P2 لا حاجة له بمعرفة نوع الكائن الذي يوجّه - يُوّشر عليه .

يمكنه أن يُوّشر (يحفظ عنوان) على كل الكائنات من الفئة الأساسية التي هي عنوان

الكائن العام الذي يكون من الكائنين (أنظر الرسم).

المصفوفة تكون بـ 800 من نوع (person) .

```
Person Arr[ ] = new Person [800] ;
```



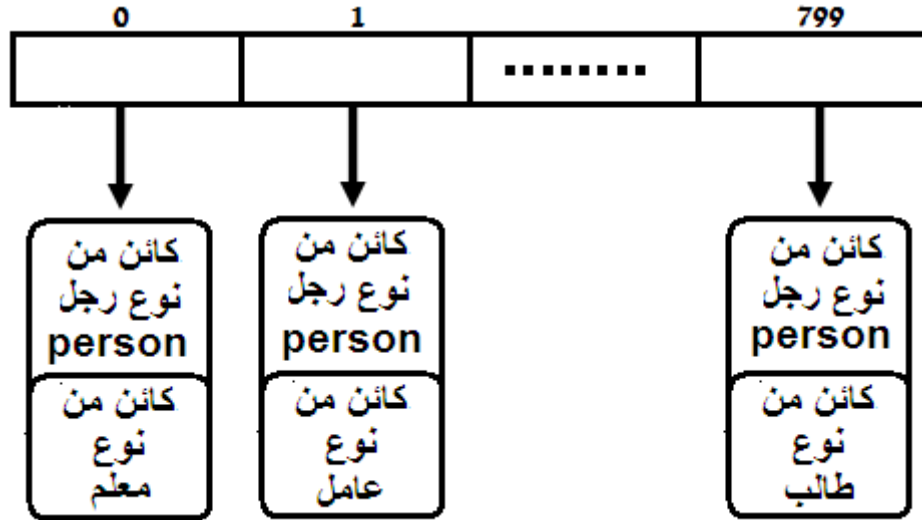
السبب أن موجه-مؤشر من الفئة الأساسية يمكنه أن يحفظ (يُوّشر على) كائن من نوع

الفئة المشتقة هو أنه عند بناء كائن من نوع الفئة المشتقة يبنى كائنين الأول من الفئة

الأساسية وبعدها كائن من الفئة المشتقة .

عند إنشاء الكائن الأول ننفذ العملية البنائية له أولاً , وبعدها عند إنشاء الكائن الثاني ننفذ العملية البنائية التابعة له.

إذاً من الممكن شرح المبنى العام للكائنات وكيفية تكون الكائنات بالصورة التالية :



### ملاحظة مهمة :

الكائن هو كائن من الفئة المشتقة لكن هو مؤلف من كائنين , الأمر New تعيد عنوان الكائن الذي يُبنى والعنوان المعاد هو عنوان الفئة الأساسية . هذا السبب هو أن كائن من نوع الفئة الأساسية يمكنه أن يوجّه على كائن من نوع الفئة المشتقة .

سؤال : إذا اردنا استعمال دوال- طرق- إجراءات موجودة داخل الفئة المشتقة ؟ كيف نستطيع فعل ذلك ؟

جواب : طريقة أ , بواسطة دالة الوهمية (virtual function)

يمكن أن نسميها إعادة كتابة من جديد للدالة .

طريقة ب , بواسطة تحويل (casting)



## الدالة الوهمية (virtual function) :

هي دالة موجودة في الفئة الأساسية لكن يمكن إعادة كتابتها وصياغتها في الفئة المشتقة. ما يحدث هو أن الفئة المشتقة تستبدل (Override) التطبيق الموجود في الفئة الأساسية بتطبيق آخر لديها .

عند استدعاء دالة وهمية ما يحدث هو أن الكائن هو الذي يُفعّل الدالة وليس المُوجّه. هذه الدالة لا يمكن أن تكون static ولا private .

في دالة عادية المُوجّه هو الذي يُفعّل الدالة, نوع المُوجّه هو الذي يقرر أي دالة- طريقة تُفعل .

دالة وهمية , الكائن هو الذي يستدعي الدالة (هو الذي يقرر أي دالة تُفعل).

```
class Person
{
    private String last_name;
    private String first_name;
    private int ID;
    private String Address;
    private String age;

    .. .. .

    public virtual void Print()
    {
        System.out.println("Name :"+ last_name + "," + first_name);
        System.out.println("ID:" + ID);
        System.out.println("Address :"+ Address );
        System.out.println("age :"+ age);
    }
}
```

```

class Teacher extends person
{
    private String Ektisas;
    private String Degree;
    private float Bouns;

    .. .. .

    @override public void Print()
    {
        super.Print();
        System.out.println("Etmahot - " + Ektisas);
        System.out.println("Degree - " + Degree);
        System.out.println("Bounus - " + Bouns);
    }
}

class App
{
    static void main()
    {
        Person p1=new Teacher ("Sami", "Salih", "5433434"...);
        P1.Print();
    }
}

```

في لغة الجافا أيضاً يمكن أن يكون هناك عدة طرق في فصيلة واحدة لها نفس الاسم ولكن المعاملات الظاهرة explicit parameters للطرق مختلفة وهذا ما يسمى بطرق التحميل الذاتي overloaded methods ومثال ذلك المنشآت دائماً لها نفس الاسم ولكن تختلف المعاملات وحينئذ يختار المترجم compiler الطريقة المناسبة عند ترجمة البرنامج طبقاً لتطابق المعاملات الظاهرة explicit parameters للطريقة المستدعاة هناك فرق بين تعدد الأشكال polymorphism والتحميل الزائد overload وهو أن في حالة التحميل الزائد overload المترجم يحدد الطريقة أثناء ترجمة البرنامج أي قبل تنفيذ البرنامج وهذا الاختيار للطريقة يسمى رابط مبكر early binding ولكن في حالة اختيار الطريقة التي تطابق نوع المعامل الضمني كما في حالة الطريقة deposit التي تم تحليلها سابقاً فإن المترجم compiler لا يأخذ أي قرار عند ترجمة البرنامج ويتم تنفيذ البرنامج قبل أن يعرف ماذا يخزن في المتغير other ولذلك فإن الآلة التخليية virtual machine وليس المترجم compiler هي التي تختار الدالة المناسبة وهذا الاختيار يسمى الرابط المتأخر late binding

## تحويل موجّه من الفئة الأساسية Base الى موجّه من الفئة المشتقة Derived : (casting)

المشكلة أن الموجّه - المؤشر من الفئة الأساسية لا يمكنه أن يُفعل دوال-طرق-إجراءات موجودة في الفئات المشتقة .

يمكن حل هذه المشكلة عن طريق تحويل الموجّه- المؤشر من موجّه- مؤشر من نوع الفئة الأساسية الى موجّه - مؤشر من نوع الفئة المشتقة حسب نوع الكائن. هذا التحويل, تحويل يمكنه أن يكون غير ممكن أو غير صحيح إذ يجب أن نقوم ببعض الفحص قبل عملية التحويل.

تحويل نمط أو نوع الموجّه- المؤشر الى الأعلى (في سلسلة الوراثة) وتسمى هذه العملية UpCasting. التحويل هذا دائماً يكون صحيح . بينما تحويل نوع الموجّه- المؤشر الى الأسفل أو ما يسمى DownCasting فهي ليست دائماً صحيحة وإنما فقط في الحالة التي يكون فيها المؤشر يُوْشِر على كائن من نوع الفئة الوارثة التي نريد أن نحول نمط الكائن إليها .

مثال :

```
Person p1 = new Student(...);
Person p2 = new Employee(...);
```

المؤشر p1 هو من نوع person ولكنه يُوْشِر على كائن من نوع الفئة Student. هذا التحويل يكون بشكل ضمني (Implicit Casting)

عملية تحويل المؤشر تكون بالصورة التالية:

```
Student S1 = (Student) p1;
Employee S1 = (Employee) p2;
```

المؤشر p1 هو من نوع person ولكنه يُوْشِر على كائن من نوع الفئة Student ولهذا يمكننا تحويل قيمة المؤشر الى قيمة من نوع Student. عملية التحويل يجب أن تكون بشكل صريح (Explicit Casting) . بكلمات أخرى يجب كتابة أسم الفئة بين قوسين هذه تسمى (DownCasting) .

لغة الجافا تعرّف الأمر instanceof الذي بواسطته يمكن أن تتم عملية التحويل .

### مثال:

```
Person p1 = new Student(...);
Student s1 = (Student) p1;

Person p2 = new Employee(...);
Employee s2 = (Employee) p2;
```

الأسطر التالية صحيحة والمؤشرات s1, s2 يمكنها أن تفعل الدوال – والطرق الموجودة داخل الفئات المشتقة Student.

لكن من الممكن أن تكون لدينا أخطاء بعملية التحويل .

```
Person p1 = new Student (....);
Employee s1 = (Employee) p1;
```

المؤشر p1 يُوْشر على عنوان كائن من الفئة Student وليس Employee. هذه عملية التحويل خاطئة.

ولهذا يجب القيام بفحص نوع نمط الكائن قبل عملية التحويل بواسطة الامر instanceof .

في المثال التالي نقوم بالفحص قبل التحويل.

```
Person p1 = new Student (...);
if ( p1 instanceof Student)
{
    Student s1 = (Student) p1;
    s1.PrintStudent();
}

Person p2 = new Employee (...);
if ( p2 instanceof Employee)
{
    Employee s2 = (Employee) p2;
    s2.PrintEmployee();
}
```

فقط مؤشر من الفئة Student, يمكنه أن يُفَعِّل دوال – طرق موجودة في الفئة Student.

فقط مؤشر من الفئة Employee, يمكنه أن يُفَعِّل دوال – طرق موجودة في الفئة Employee.

## الفئة Object:

الفئة Object عبارة عن فئة توفرها لغة الجافا وتجعلها بشكل تلقائي ال Super , لجميع الفئات, معنى ذلك أن أي فئة جديدة نكتبها ترث الفئة Object. هي تحوي عدد صحيح من الدوال – الطرق بعضها وهمية التي تعني شيئاً لجميع الكائنات مثل الدالة toString التي يمكن استخدامها للحصول على وصف حالة الكائن.

### public boolean equals(Object obj)

هذه الدالة কিفما هي مطبقة داخل الفئة Object, تفحص هل المؤشرين يؤشران على نفس الكائن. تطبق هذه الفئة من جديد داخل الفئة المشتقة لكي نفحص فحوى الكائنين .

### public virtual string ToString ()

هذه الدالة কিفما هي مطبقة داخل الفئة Object, تعيد إسم الفئة, تطبق هذه الفئة من جديد داخل الفئة المشتقة لكي تعيد لنا وصف لقيم الميزات কিفما نريد .

الهندسة للنشر

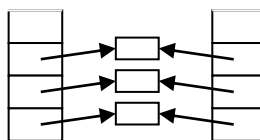
## سؤال في تعدد الأشكال : Polymorphism

أمامك واجهة تطبيق للفئة راصّة - Stack , التي تعرّف مبنى معطيات كبير غير محدود , يمكن تخزين داخل الراصة كائنات من نوع الفئة Object .  
الراصّة كما هو معروف , لديها آلية تسمى (LIFO) أي من يدخل أولاً الى الراصة يخرج أخيراً .  
أمامك واجهة التطبيق للفئة Stack:

Stack()	عملية بنائية تبني راصة فارغة
bool Empty()	تعيد "صدق" (true) اذا كانت الراصة فارغة خلاف ذلك تعيد الدالة "كذب" (false)
Object Peek()	تعيد الكائن الموجود في رأس الراصة بدون إخرجه من الراصة .
Object Pop()	تخرج الكائن الموجود في رأس الراصة وتعيده .
void Push (Object item)	تتلقى كائن وتدخله الى رأس الراصة

(أ) الفئة MyStack ترث من الفئة Stack وتعرّف بها عمليّتين - دالتين :

- البناء الافتراضي .
- البناء الناسخ , يقوم البناء الناسخ بنسخ راصّه من نوع MyStack بالطريقة المشروحة في الرسمة التالية. كل مؤشر (موجّه) في الراصة المنسوخة يؤشر على الكائن الذي يؤشر عليه المؤشر الموازي في الراصة الأصلية :
- iii.



الراصّة الأصلية الراصة المنسوخة (الراصّتين من نوع MyStack)

عرّف الفئة وطبق (أكتب) العمليّتين السابقتين .

ب) طبق الدالة داخل الفئة , بحيث أنها تعيد نص يصف كل أعضاء الراسّة , من المكان الاول حتى الأخير . كل خلية تعرض حسب نوعها . لا يجب أن نغيّر بالفئة للرّاسّة ولا نريد أن نستعمل الدالة

**void Push (Object item )**

ج) أظهر الاسطر الغير صحيحة في المقطع التالي . إشرح الخطأ

```
public static void main(String[] args)
{
    (1) Stack s1 = new MyStack();
    (2) Stack s2 = new Stack (s1);
    (3) MyStack s3 = new MyStack (s1);
    (4) MyStack s4 = new Stack();
    (5) MyStack s5 = new MyStack();
    (6) MyStack s6 = (MyStack) s2;
}
```

**الحل :**  
(أ)

```
public class MyStack extends Stack
{
    public MyStack()
    {
    }
    public MyStack (MyStack myStack)
    {
        Stack tmp = new Stack();
        while ( ! myStack.empty() )
        {
            tmp.push (myStack.pop());
        }
        while (!tmp.empty())
        {
            Object obj = tmp.pop();
            this.push (obj);
            myStack.push (obj);
        }
    }
}
```

(ب)

```

public String toString()
{
    String str = "";
    MyStack tmp = new MyStack (this);
    while ( ! tmp.empty() )
    {
        str += " " + tmp.pop() + " ";
    }
    return str;
}

```

(ج)

الواامرفي السطر 1 والسطر 5 صحيحة

الواامرفي السطر 2 غير صحيحة, محاولة إستعمال عملية بنائية غير موجودة في الفئة الاصلية

الواامرفي السطر 3 غير صحيحة, حتى نجعلها صحيحة كان يجب أن نعمل تحويل المؤشر S1 الى الاسفل عودة الى نوع النمط MyStack و عندها كان من الممكن إرسال المؤشر S1 كبارمتر مناسب للعملية البنائية الناسخة .

الواامرفي السطر 4 غير صحيحة , لا يمكن أن نقوم بهذه عملية التحويل . كائن من الفئة الاساسية يؤشر على مؤشر من نوع MyStack من الفئة العليا .

الواامرفي السطر 6 غير صحيحة , لا يمكن أن نقوم بهذه عملية التحويل الى الاسفل لكائن من نوع الفئة العليا . كائن من الفئة الاساسية يؤشر على مؤشر من نوع MyStack من الفئة العليا .



# معالجة الاستثناءات

## Exception Handling

نسخة تجريبية - ليست للنشر



# معالجة الاستثناءات

## Exception Handling

الإستثناء هو مؤشر لحدوث خطأ أثناء عملية تنفيذ البرنامج مما يؤدي إلى تعطيل التسلسل الطبيعي لتعليمات البرنامج وقد تعلمنا في الفصول السابقة أن الوراثة في لغة الجافا تعطيها صفة الإمتدادية وهذه الصفة يمكن أن تزيد من عدد ونوع الأخطاء التي يمكن أن تحدث حيث إن كل فئة جديدة تضاف إلى البرنامج يمكن أن تضيف مصدرا من مصادر الاستثناءات في البرنامج.

إذاً نستطيع القول أن الاستثناء هو حدوث خطأ ما وهذا الخطأ ليس خطأ في بناء الجملة syntax error ولكنه قد يكون له العديد من المصادر مثل القسمة على صفر ومعاملات غير متاحة للدالة و الإشارة إلى عنصر في المصفوفة خارج نطاقها.

عند حدوث إستثناء يحتاج البرنامج الى معالجة هذا الإستثناء لكي يستمر تنفيذ البرنامج بصورة طبيعية .

### أساسيات معالجة الإستثناء في لغة الجافا :

لقد أدت مشاكل استخدام شفرة الأخطاء Error codes إلى تطوير آلية جديدة لمعالجة الاستثناءات في لغة الجافا تعتمد على الكائنات مما أدى إلى برامج سهلة القراءة والتتبع وكذلك برامج أكثر مرونة.

وفي هذا النموذج عند حدوث استثناء أثناء تشغيل برنامج الجافا إما البرنامج program أو آلة لغة الجافا الافتراضية JVM تنشئ كائن لوصف الاستثناء ويشمل هذا الكائن قيم المتغيرات في لحظة حدوث الاستثناء.

إذا تم إنشاء الكائن من البرنامج فإن البرنامج يمرر ذلك الكائن إلى آلة الجافا الافتراضية JVM وعند استقبال الكائن تبحث في البرنامج عن معالج الاستثناء exception handler الذي يمكن أن يعالج الاستثناء الموصوف بالكائن. إذا وجد المعالج يتم تمرير الكائن لمعالج الاستثناء الذي يقوم باستخدام محتويات الكائن لمعالجة الاستثناء. إذا لم يوجد معالج الاستثناء يتوقف البرنامج عن التنفيذ.

### مثال 1:

الصورة التالية تبين برنامج بسيطاً لقسمة رقمين إذا تتبعنا هذا البرنامج نجد أنه يتم تنفيذ طريقة صحيحة لأنه يوجد قيمة غير صفرية للمقام (d).

The screenshot shows the JCreator IDE with a project named 'exceptions'. The main class is 'EXCP1'. The code in the editor is as follows:

```

1 public class EXCP1
2 {
3     public static void main(String args[])
4     {
5         int d=1;
6         int a=60/d;
7         System.out.println("a="+a);
8         System.out.println("After Exception");
9     }
10 }
11

```

The IDE's output window at the bottom shows the following text:

```

-----Configuration: JDK version 1.3 <Default>-----
Process completed.
C:\Program Files\Xinox Software\JCreator\GE2001.exe
a=60
After Exception
Press any key to continue...

```

The status bar at the bottom indicates 'Ln 5, Col 10' and 'DOS'.

لكن الشكل التالي يبين نفس البرنامج ولكن تم إعطاء المقام قيمة صفرية ولذلك اذا تتبعنا الشكل نجد أن المترجم (compiler) قد أنهى ترجمة البرنامج بنجاح وهذا ملاحظ من الجملة process completed ولكن عند تنفيذ البرنامج هناك خطأ تنفيذي (استثناء) وهو القسمة على صفر فقد بحث عن معالج للخطأ في البر بامج فلم يجد ولذلك أعطى رسالة

Exception in thread "main" java.lang.ArithmeticException:  
/by zero at Excp1.main <Excp1.java:6>

وهذا يبين أن كائن الاستثناء من الفصيلة ArithmeticException ويشمل بيان الخطأ وهو القسمة على صفر (/by zero) وقد بحث عن معالج داخل البرنامج فلم يجد وتلاحظ في شكل التالي إنهاء تنفيذ البرنامج قبل تنفيذ جمل الطباعة

The screenshot shows the JCreator IDE with a project named 'exceptions'. The main class is 'EXCP1'. The code in the editor is as follows:

```
1 public class EXCP1
2 {
3     public static void main(String args[])
4     {
5         int d=0;
6         int a=60/d;
7         System.out.println("a="+a);
8         System.out.println("After Exception");
9     }
10 }
11
```

The output window shows the following message:

```
Process completed.
C:\Program Files\Xinox Software\JCreator\GE2001.exe
Exception in thread "main" java.lang.ArithmeticException: / by zero
at EXCP1.main(EXCP1.java:6)
Press any key to continue...
```

مثال 2:

يبين الشكل التالي برنامج بسيطاً لتعريف مصفوفة مكونة من عنصرين من نوع الأرقام الصحيحة وإعطاء قيم للعنصرين وطباعة قيم العنصرين . تلاحظ من الشكل تنفيذ البرنامج بصورة طبيعية وتم إعطاء نتائج الطباعة.

ولكن الشكل الذي يليه يبين نفس البرنامج مع إضافة الجملة `a[2]=6` ونتيجة أن هذا العنصر خارج نطاق تعريف المصفوفة فقد وقع استثناء وتم البحث عن معالج داخل البرنامج فلم يوجد، فتوقف تنفيذ البرنامج قبل تنفيذ جمل الطباعة وأعطى رسالة الخطأ التالية :

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException at Excp1.main <Excp3.java:9>

وهذا يبين أن كائن الاستثناء من الفصيلة `ArrayIndexOutOfBoundsException`

The screenshot shows the JCreator IDE with a project named 'exceptions'. The main class is 'Excp3'. The code in 'Excp3.java' is as follows:

```

1 public class Excp3
2 {
3     public static void main(String args[])
4     {
5
6         int a[]=new int[2];
7         a[0]=2;
8         a[1]=4;
9
10        System.out.println("a[0]="+a[0]);
11        System.out.println("a[1]="+a[1]);
12    }
13 }
14

```

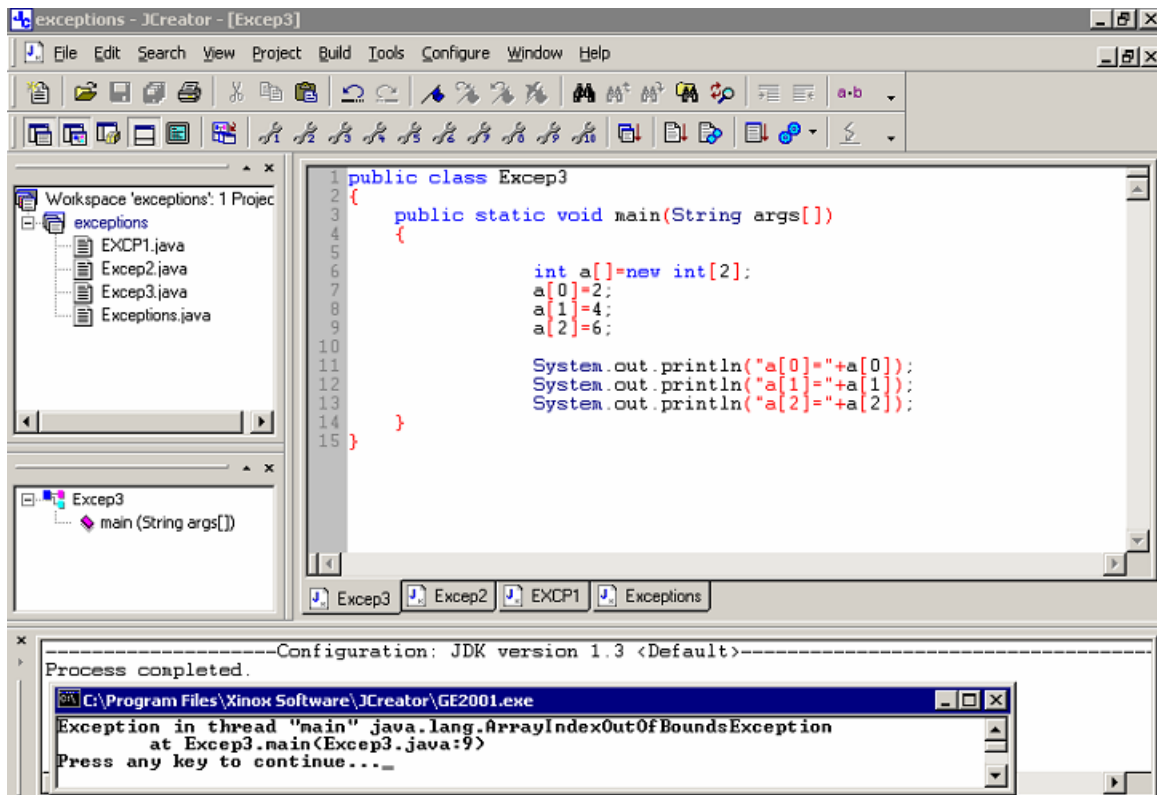
The output window shows the following message:

```

-----Configuration: JDK version 1.3 <Default>-----
Process completed.
C:\Program Files\Xinox Software\JCreator\GE2001.exe
a[0]=2
a[1]=4
Press any key to continue...

```

وقع استثناء وتم البحث عن معالج داخل البرنامج فلم يوجد، فتوقف تنفيذ البرنامج قبل تنفيذ جمل الطباعة وأعطى رسالة خطأ:



```
1 public class Excep3
2 {
3     public static void main(String args[])
4     {
5
6         int a[]=new int[2];
7         a[0]=2;
8         a[1]=4;
9         a[2]=6;
10
11         System.out.println("a[0]="+a[0]);
12         System.out.println("a[1]="+a[1]);
13         System.out.println("a[2]="+a[2]);
14     }
15 }
```

Process completed.

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException  
at Excep3.main(Excep3.java:9)  
Press any key to continue...

الهندسة للنشر

## أنواع الاستثناءات Exception Types

من الأمثلة السابقة تبين أن هناك العديد من أنواع الاستثناءات ومن معرفتنا للغة الجافا بأنها تتكون من فئات فإن الاستثناءات في الجافا هي فئات `classes` وكل فصيلة `class` تختص بنوع من الاستثناءات وجميع هذه الفئات ترث الفئة العليا `Throwable` وتوجد فئتان فرعيتان ترثان هذه الفئة وهما `Exception subclass` و `Error subclass` وهذه الفئات موجودة في الحزمة `java.lang` وهذه الفئات الفرعية تصنف الاستثناءات أي ذات علاقة بالبرنامج `program related` أم هي ذات علاقة بآلة الجافا الافتراضية `JVM`.

الجدول التالي يصف بعض الفئات الفرعية للفئة `Exception`

<code>ArithmeticException</code>	الكائن المنشئ من هذه الفئة يصف مشكلة حسابية مثل محاولة القسمة على صفر
<code>ArrayIndexOutOfBoundsException</code>	الكائن المنشئ من هذه الفئة يصف محاولة الوصول الى عنصر في المصفوفة وهو غير معرف (غير موجود)
<code>ClassNotFoundException</code>	الكائن المنشئ من هذه الفئة يصف محاولة الوصول الى تحميل ملف فئة غير موجودة.
<code>FileNotFoundException</code>	الكائن المنشئ من هذه الفئة يصف محاولة ملف غير موجود
<code>IOException</code>	الكائن المنشئ من هذه الفئة يصف خطأ عاماً أثناء عملية إدخال أو أخراج وتوجد فئة فرعية من هذه الفئة لتصف الخطأ بدقة
<code>NullPointerException</code>	الكائن المنشئ من هذه الفئة يصف محاولة استدعاء طريقة متغير كائن ليس له مرجعية لأي كائن <code>null reference</code>

## معالجة الاستثناءات في الجافا Exception Handling In java

تتم معالجة الأخطاء في لغة الجافا باستخدام مجموعة من التعليمات وهي :

- a- try block
- b- catch blocks
- c- finally block
- d- throw statement
- e- throws clause

أولاً : try...catch finally blocks

تستخدم لغة الجافا التعليمة try لتحديد الجزء من البرنامج الذي يحتمل أن يحدث به خطأ ويجب أن يتبع هذا الجزء مباشرة تعليمة catch أو أكثر لتحديد طريقة معالجة الأنواع المتوقعة من الاستثناءات ثم يتبع آخر تعليمة catch تعليمة finally وهي اختيارية وتستخدم لتحديد جزء من البرنامج يجب تنفيذه بغض النظر هل حدث استثناء في جزء تعليمة try أم لم يحدث استثناء ويكون الشكل العام للتعامل مع الاستثناءات في لغة الجافا كالتالي:

```
try
    // Tested statements

catch ( ExceptionType 1   exob1)
    // exception handler for ExceptionType1

catch ( ExceptionType 2   exob2)
    // exception handler for ExceptionType2

finally
    // Statements that must be executed
```



### مثال 3 :

الشكل التالي يبين نفس البرنامج , قسمة رقمين الذي تم شرحه في مثال سابق وفي هذا الشكل يتم معالجة الاستثناء ArithmeticException الذي حدث عندما أعطى المقام d قيمة صفرية ويتم ذلك باستخدام التعليمة الأمر لإحتواء الجمل التي تسببت في الاستثناء كالتالي:

```
try
    int d=0;
    int a=60/d;
```

ويتبع ذلك تعليمة catch التي تعالج هذا النوع من الاستثناء كالتالي :

```
catch (ArithmeticException e)
    System.out.println("divide by zero");
    System.out.println(e.getMessage());
```

ويمكن تتبع تنفيذ البرنامج حيث تم حدوث استثناء عند تنفيذ السطر رقم 8 في الشكل التالي وهو محاولة القسمة على صفر فتم إنشاء كائن من الفئة ArithmeticException وتم البحث عن معالج لهذا الاستثناء في البرنامج فوجد معالج مطابق للاستثناء في سطر 10 فتم انتقال تسلسل تنفيذ البرنامج إلى معالج الاستثناء الذي يحتوي على جملتين الأولى في السطر 13 لإظهار الرسالة divide by zero والثانية في السطر 14 لإظهار الرسالة التي يحتويها كائن الاستثناء e وهي by zero / وبعد تنفيذ المعالج تم تنفيذ الجملة التي تتبع المعالج مباشرة في سطر 16 لإظهار الرسالة After Exception Handling وتم إنهاء البرنامج بصفة طبيعية.

الصورة التالية تبين نفس برنامج الذي تم شرحه في مثال سابق , وفي هذا الشكل يتم معالجة الاستثناء ArrayIndexOutOfBoundsException الذي حدث عند إعطاء عنصر المصفوفة a[2] قيمة 6 . يتم التقاط الاستثناء باستخدام الأمر try لإحتواء الجمل التي تسببت في الإستثناء كالتالي :

```
try
    a[0]=2;
    a[1]=4;
    a[2]=6;
```

The screenshot shows the JCreator IDE with a project named 'exceptions'. The main class is 'Excep2'. The code in the editor is as follows:

```

1 public class Excep2
2 {
3     public static void main(String args[])
4     {
5         try
6         {
7             int d=0;
8             int a=60/d;
9         }
10        catch (ArithmeticException e)
11        {
12            System.out.println("divide by zero");
13            System.out.println(e.getMessage());
14        }
15        System.out.println("After Exception Handling");
16    }
17 }
18

```

The output window shows the following text:

```

Process completed.
C:\Program Files\Xinox Software\JCreator\GE2001.exe
divide by zero
/ by zero
After Exception Handling
Press any key to continue...

```

الصورة التالية يتم معالجة الاستثناء `ArrayIndexOutOfBoundsException` عند إعطاء عنصر المصفوفة `a[2]` قيمة 6

The screenshot shows the JCreator IDE with a project named 'exceptions'. The main class is 'Excep3'. The code in the editor is as follows:

```

1 public class Excep3
2 {
3     public static void main(String args[])
4     {
5         int a[]=new int[2];
6         try
7         {
8             a[0]=2;
9             a[1]=4;
10            a[2]=6;
11        }
12        catch (ArrayIndexOutOfBoundsException e)
13        {
14            System.out.println("Array Index out of bounds..");
15        }
16        finally
17        {
18            System.out.println("The code here must be executed");
19        }
20    }
21 }

```

The output window shows the following text:

```

Process completed.
C:\Program Files\Xinox Software\JCreator\GE2001.exe
Array Index out of bounds..java.lang.ArrayIndexOutOfBoundsException
The code here must be executed
Press any key to continue...

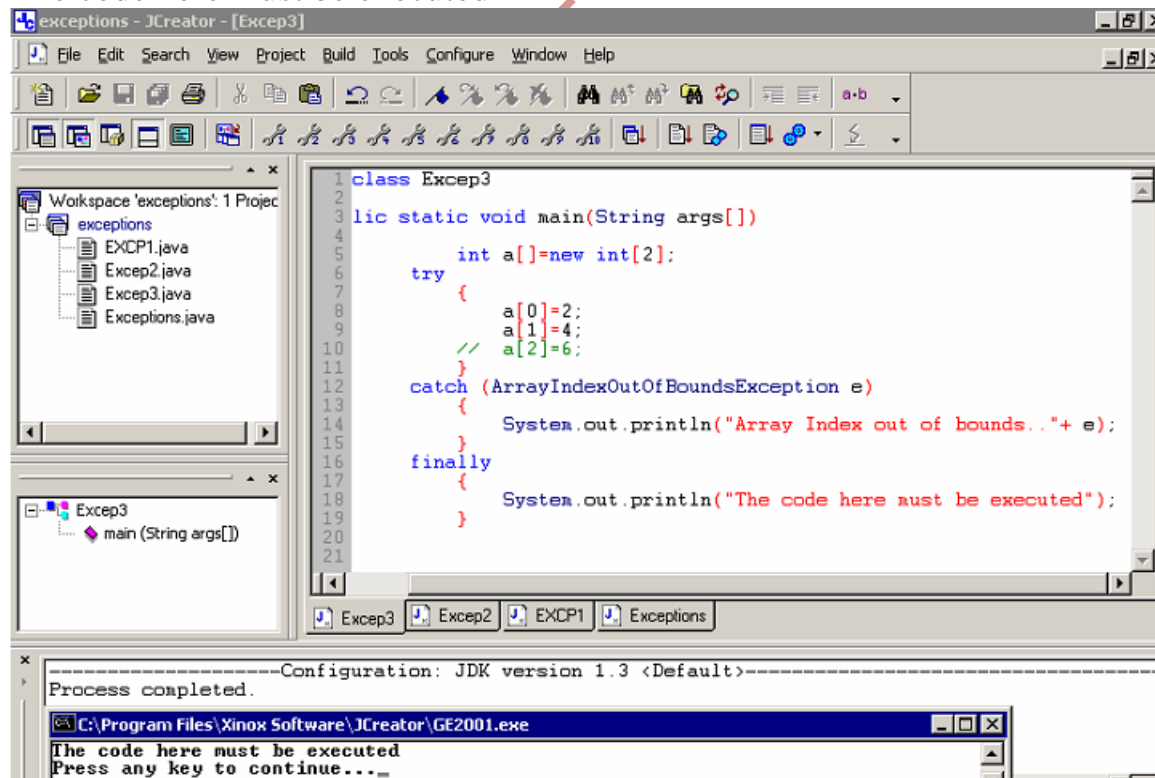
```

ولمعالجة الاستثناء `ArrayIndexOutOfBoundsException` نستخدم الأمر `catch` كالتالي:

```
catch (ArrayIndexOutOfBoundsException e)
    System.out.println("Array Index out of bounds.." + e);
```

وفي هذا البرنامج تم استخدام الأمر `finally` لإيضاح كيف يمكن إحتواء جزء البرنامج الذي يجب تنفيذه بعد الأمر `try` سواء حدث إستثناء أم لا. وبتتبع تنفيذ هذا البرنامج في الشكل السابق نجد أنه تم تنفيذه بطريقة صحيحة حيث تم النقاط الإستثناء عند تنفيذ السطر 10 وتم البحث عن معالج الإستثناء وبدأ تنفيذه من السطر 12 وفيه تم إظهار الرسالة `Array Index out of bounds` بالإضافة إلى الذي يحتويها هو الكائن (e) وهو `java.lang.ArrayIndexOutOfBoundsException` ثم بعد ذلك تم تنفيذ جزء البرنامج الذي تحويه التعليمة وهو إظهار الرسالة :

The code here must be executed



الصورة السابقة تبين تنفيذ البرنامج السابق بعد إلغاء الجملة التي تسببت في الاستثناء فنجد أنه تم تنفيذ الجملة داخل الامر `finally` كما ذكرنا سابقاً أنه يتم تنفيذ هذا الجزء من البرنامج سواء كان هناك استثناء أم لا .

عند توقع حدوث أكثر من إستثناء في جزء من البرنامج فيمكن إحتواء هذا الجزء بالامر `try` لالتقاط الأنواع المختلفة من الاستثناءات ثم نتبع ذلك بالامر `catch` كلاً منها يعالج نوعاً من أنواع الإستثناءات المتوقعة.

نسخة تجريبية - ليست للنشر

# الواجهات -

# Interfaces

المهندس مؤنس قشقوش - نحف

# الواجهات - Interfaces

## الفئة المجردة بلغة الجافا Abstract Class in java

تستخدم الفئات المجردة للجافا "Java Abstract classes" للإعلان عن الخصائص المشتركة للفئات الفرعية "subclasses". الفئة المجردة لا يمكن أن تكون فئة مثل "Instance". لا يمكن استخدامها إلا بوصفها الفئة المتفوقة "superclass" للفئات الأخرى اللاتي إمتدّن من الفئة المجردة. يتم تعريف الفئات المجردة بالكلمة المجردة "abstract keyword". وتستخدم الفئات المجردة لتوفير قالب أو تصميم لفئات فرعية محددة أسفل شجرة الوراثة "inheritance tree".

مثل أي فئة أخرى، يمكن لفئة مجردة أن تحوي حقول التي تصف الخصائص والأساليب التي تصف الإجراءات التي يمكن أن تؤديها الفئة. يمكن لفئة مجردة "abstract class" أن تشمل الطرق التي لا تحتوي على التنفيذ "implementation". وتسمى هذه بالطريقة-الدالة التجريدية "abstract methods". يجب الإعلان نهاية الطريقة التجريدية بفاصلة منقوطة ";" "semicolon" بدلا من كتلة. إذا كانت الفئة لديها أي طريقة مجردة "abstract methods"، سواء المعلنة أو الموروثة، فيجب أن تعلن الفئة كاملة كفئة مجردة. و تستخدم الطريقة التجريدية "abstract methods" لتقديم نموذج للفئات التي ترث المنهج المجرد "abstract methods". لا يمكن أن يكون للفئات المجردة مثل، ويجب أن يكونوا كفئات فرعية ، ويجب توفير التطبيقات الفعلية للطرق التجريدية "abstract methods". و بطبيعة الحال يمكن ان يتم تجاوز أي تنفيذ محدد، من قبل فئات فرعية إضافية "subclasses". يجب أن يكون للكائن تنفيذ لجميع طرقه. تحتاج إلى إنشاء فئة فرعية لتوفر تنفيذ للطرق-دوال التجريدي "abstract method".

ويمكن لفئة مجردة تسمى بـ "Vehicle" أن يتم تحديدها مجردة لتمثيل التجريد العام  
لوسيلة نقل "Vehicle"، كما أن خلق مثيلات "instances" من الفئة لا يكون ذا مغزى

```
abstract class Vehicle
{
    int numofGears;
    String color;
    abstract boolean hasDiskBrake();
    abstract int getNoofGears();
}
```

مثال على فئة تسمى شكل "shape" تعتبر كفئة مجردة :

```
abstract class Shape
{
    public String color;

    public Shape()
    {
    }

    public void setColor(String c)
    {
        color = c;
    }

    public String getColor()
    {
        return color;
    }

    abstract public double area();
}
```

يمكننا أيضا تنفيذ الفئة أشكال عامة "shapes" كفئة مجردة حتى نتمكن من رسم خطوط ودوائر ومثلثات الخ ، جميع الأشكال لها بعض الحقول المشتركة والطرق ، ولكن لكل واحد منها ، بطبيعة الحال ،المزيد من الحقول والطرق. الفئة مجردة تضمن بأن يكون لكل شكل نفس الخصائص الأساسية. ونقوم بإعلان هذه الفئة مجردة لأنه لا يوجد شيء يمثل شكل عام. يمكن أن يكون هناك فقط الأشكال الملموسة مثل مربعات مثلثات ودوائر الخ ....

```
public class Point extends Shape
{
    static int x, y;
    public Point()
    {
        x = 0;
        y = 0;
    }
    public double area()
    {
        return 0;
    }
    public double perimeter()
    {
        return 0;
    }
    public static void print()
    {
        System.out.println("point: " + x + "," + y);
    }
    public static void main(String args[])
    {
        Point p = new Point();
        p.print();
    }
}
```

Output  
point: 0, 0

لاحظ أنه ، من أجل إنشاء كائن نقطة "Point" ، ففئتها لا يمكن أن تكون مجردة. هذا يعني أن كل المناهج المجردة للفئة شكل "Shape" يجب أن تنفذ من قبل الفئة نقطة "Point".



يجب أن تُعرف الفئة الفرعية التنفيذ لكل طريقة - دالة تجريدية "abstract method" للفئة المتفوقة المجردة ، أو الفئة الفرعية نفسها سوف تكون أيضا مجردة. وبالمثل يمكن أن يتم إنشاء كائنات أخرى باستخدام الشكل العام للفئة التجريدية. العيب الكبير في استخدام الفئات المجردة هو عدم القدرة على استخدام وراثتها متعددة. بمعنى، عندما تمتد فئة من فئة مجردة، فإنه لا يمكنها تمديد أية فئة أخرى.

## الواجهات بلغة الجافا Java Interface

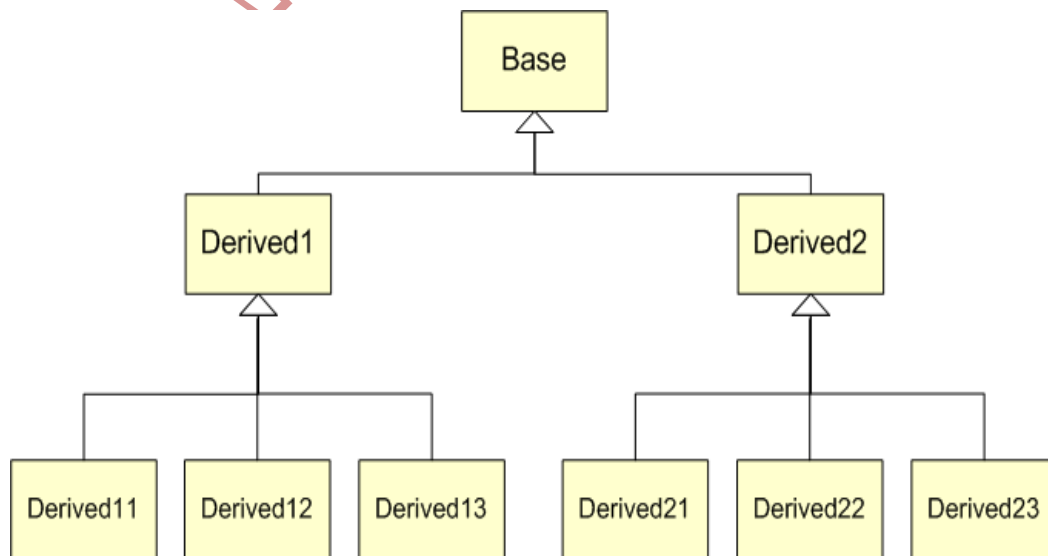
### لماذا الواجهات ؟

نعرض مشكلة تواجه المبرمجين بلغات برمجة الكائنات الموجهة

### المشكلة الاولى:

بسبب عملية الوراثة الفئة المشتقة تُجبر على أخذ كل الميزات من الفئة الموروثة وأيضاً جميع الدوال. في بعض الحالات نحتاج أن نُجبر فئات معينة أن تبني دوال - طرق خاصة بها أي أن يكون بها دوال معينة محددة خاصة بها .

مثال:



نفترض أن الفئات Derived11, Derived13 و Derived21 نريدها أن تحوي (تكتب) بداخلها دالة- طريقة تمكّن طباعة المعلومات كتقارير على شكل صفحات HTML . ولكن لا حاجة على الفئات الأخرى أن تضم هذه الطريقة-الدالة .

#### إقتراحات حلول :

وضع هذه الدالة في الفئة Base لا يمكن أن يكون حل منطقي , لأن جميع الدوال الأخرى يمكنهم أن تستعمل هذه الدالة – الطريقة . لا يمكن بناء فئة Base أخرى ووضع بها هذه الطريقة – الدالة ثم وجعل جميع الفئات Derived11, Derived13 و Derived21 ترث هذه الفئة لأن لغة الجافا لا تدعم الوراثة المتعددة "Multiple Inheritance" .

#### سؤال:

إذاً كيفية إجبار مجموعة من الفئات الموجودة داخل رسم تخطيطي هرمي (كالموجود في الصفحة السابقة) على تنفيذ دوال- طرق معينة , بينما لا نريد إجبار فئات أخرى التي هي أيضاً موجودة داخل نفس رسم التخطيط الهرمي على تنفيذ هذه الدوال ؟؟

**المشكلة الثانية :**

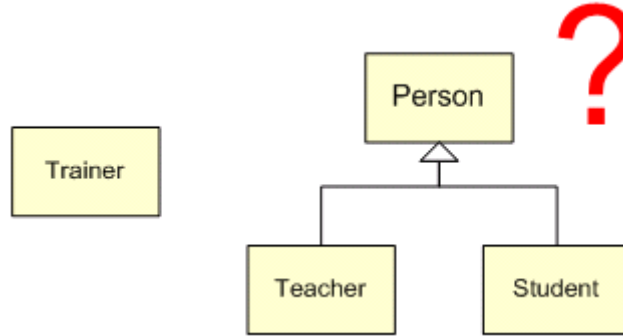
لبرمجة برنامج لإدارة كلية أو جامعة نحتاج الى تعريف كم هائل من الدوال منها :

Teacher – كل ما يتعلق بتفاصيل محاضر – أستاذ , كل ما يتعلق بتعيين الحصص لهذا المحاضر وكل ما يتعلق بالأجرة الشهرية للمحاضر.

Student – كل ما يتعلق بتفاصيل الطالب , كل ما يتعلق بتعيين الكورسات لهذا الطالب وكل ما يتعلق بالعلامات للطالب .

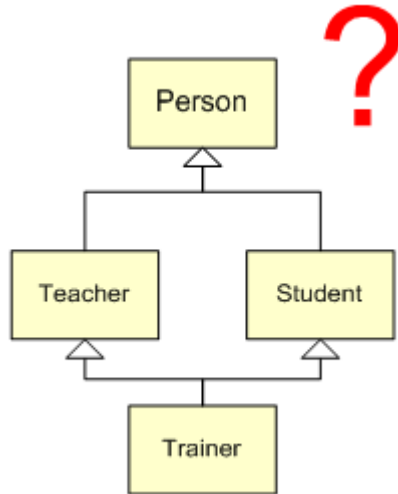
Trainer – كل ما يتعلق بتفاصيل المُعيد (مساعد محاضر) , كل ما يتعلق بتعيين الحصص لهذا المُعيد وكل ما يتعلق بالأجرة الشهرية للمُعيد.

إقتراح رسم تخطيطي هرمي :

**لكن ماذا عن الفئة Trainer ?**

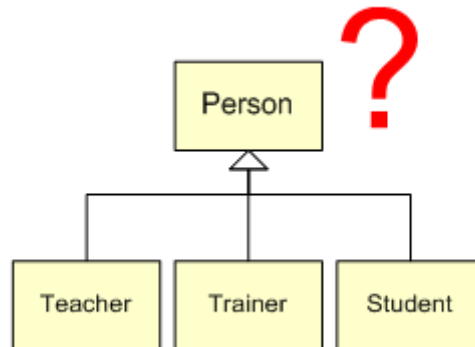
**الإحتمال الاول** هو أن الفئة Trainer ترث الفئة Teacher وأيضاً ترث الفئة Student , لأن المُعيد هو طالب وهو أيضاً محاضر Trainer is a Teacher و – Trainer is a Student .

في هذه الحالة الفئة Trainer, ترث الدالة – الطريقة "حساب أجرة شهرية" من الفئة Teacher وترث الدالة – الطريقة "معالجة العلامات" من الفئة Student .



هذا الاحتمال لا يمكن تطبيقه بلغة الجافا،  
بسبب أن الجافا لا تدعم الوراثة المتعددة  
"Multiple Inheritance"

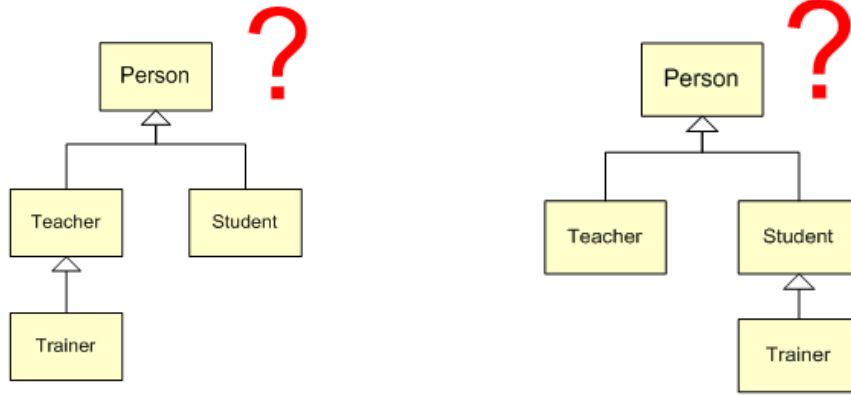
الإحتمال الثاني: هو أن الفئة Trainer ترث الفئة person مثل الفئة Student والفئة Teacher .



طبعا تصبح لدينا مشكلة وهي تكرار دالة "حساب أجره شهرية" التي سوف تنطبق داخل الفئة Teacher وايضاً داخل الفئة Trainer , طبعا تصبح لدينا مشكلة أخرى وهي تكرار دالة "معالجة العلامات" التي سوف تنطبق داخل الفئة Student وايضاً داخل الفئة Trainer .

الهدف من الوراثة هو تقليل تكرار دوال - طرق وحل بهذه الطريقة ليس بمفضل.

**الإحتمال الثالث:** هو أن الفئة Trainer ترث الفئة Student أو الفئة Teacher عندها تكرار دوال يكون بشكل أقل .



أيضاً هذه الطريقة غير مفضلة .

لإعطاء حل جيد لمثل هذه المشاكل نستعمل الواجهات (Interface) .

### شرح مفصل :

في جافا، يتم حل مشكلة الوراثة "inheritance" المتعددة بإستعمال بناء قوي يسمى الواجهات "interfaces". ويمكن استخدام واجهة لتحديد القالب العام "generic template" ومن ثم فئة مجردة أو أكثر لتحديد التطبيقات جزئية للواجهة. الواجهات تحدد فقط الإعلان عن المنهج "method" (العام و الضمني و المجردة "implicitly public and abstract")، ويمكن أن تحتوي فقط على الحقول (التي هي ضمنية ، العامة ، نهائية ، ثابتة "implicitly public static final"). تعريف الواجهة يبدأ مع الكلمة الرئيسية واجهة "interface". واجهة لفئة مجردة مثل هذه لا يمكن إنشاء مثيل لها "instantiated".

الوراثة المتعددة "Multiple Inheritance" يسمح بها عند تمديد الواجهات. واجهة واحدة يمكن أن تمتد الى واجهة أو أكثر. جافا لا تدعم "Multiple Inheritance" الوراثة المتعددة ، ولكنها تسمح لك بتمديد "extend" فئة واحدة وتنفيذ العديد من الواجهات.

إذا كانت الفئة التي تُطبق الواجهة لا تُعرّف جميع (دوال) المناهج للواجهة، فيجب أن يتم إعلانها مجردة و تعريفات المناهج يجب ان تطبقها في الفئات الفرعية الاتي يمتدّن من الفئة المجردة.

## المثال 1: فيما يلي مثال لواجهة بإسم شكل "interface"

```
interface Shape
{
    public double area();
    public double volume();
}
```

يوجد أدناه الفئة نقطة "Point class" التي تطبق الواجهة الشكل "Shape interface".

```
public class Point implements Shape
{
    static int x, y;

    public Point() {
        x = 0; y = 0;
    }

    public double area() {
        return 0;
    }

    public double volume() {
        return 0;
    }

    public static void print()
    {
        System.out.println("point: " + x + "," + y);
    }

    public static void main(String args[])
    {
        Point p = new Point();
        p.print();
    }
}
```

على نحو مماثل، يمكن إنشاء الكائنات شكل الأخرى بإستعمال واجهة برمجة "interface programming" عن طريق تنفيذ واجهة شكل عامة .

## مثال 2 :

وفيما يلي برنامج لواجهات جافا "java interfaces program" يعرض قوة برمجة الواجهة "interface programming" لجافا

القائمة أدناه تظهر لنا 2 من الواجهات و 4 فئات واحدة منهم فئة مجردة.

ملاحظة : المنهج toString في الفئة A1 هو نسخة تجاوزناها للمنهج الذي تم تحديده في فئة تسمى كائن "Object". الفئات B1 و C1 تلبي متطلبات عقد الواجهة. ولكن بما أن الفئة D1 لا تقوم بتعريف كافة المناهج لتطبيق الواجهة I2 ، فإن الفئة D1 أعلنت مجردة.

أيضا ،

تُنتج "i1.methodI2" خطأ في عملية التحويل البرمجي "compilation" لأن هذا المنهج لم يعلن في I1 أو أي واحدة من الواجهات الفائقة لها هذا إذا كانت موجودة. بالتالي المسبل او ما يسمى بالتحويل الى أسفل "downcast" لمرجعية الواجهة I1 يحل المشكلة كما تظهر في البرنامج. المشكلة نفسها تنطبق على "i1.methodA1" ، و تحل ايضا عن طريق المسبل او ما يسمى بالتحويل الى أسفل "downcast".  
لتحويل

عندما نستدعي المنهج toString() الذي هو منهج للفئة كائن "Object"، لا يبدو أن هناك أي مشكلة باعتبار أن كل واجهة أو فئة تمتد من كائن "Object" و أي فئة يمكن ان تتجاوز toString() الافتراضي لتناسب مع احتياجات التطبيق الخاص بك.

((C1) O1). methodI1 ()

لها تحويل برمجي ناجح ، ولكن يُنتج ClassCastException في وقت التشغيل "runtime". هذا لأن B1 ليس لديها أي علاقة مع C1 إلا أنهم "الأشقاء". لا يمكنك قولبة الأشقاء واحدا في الآخر.

عندما يتم استدعاء منهج واجهة معينة على مرجع معين ، السلوك الذي سوف ينتج سيكون مناسب للفئة التي منها تم إنشاء هذا الكائن المعينة. هذا هو تعدد أشكال وقت التشغيل "runtime polymorphism" استنادا إلى الواجهات و هيمنة المناهج.

```
interface I1
{
    void methodI1(); // public static by default
}

-----

interface I2 extends I1
{
    void methodI2(); // public static by default
}

-----

class A1
{
    public String methodA1()
    {
        String strA1 = "I am in methodA1 of class A1";
        return strA1;
    }

    public String toString()
    {
        return "toString() method of class A1";
    }
}

-----

class B1 extends A1 implements I2
{
    public void methodI1()
    {
        System.out.println("I am in methodI1 of class B1");
    }

    public void methodI2()
    {
        System.out.println("I am in methodI2 of class B1");
    }
}

-----

class C1 implements I2
{
    public void methodI1()
    {
        System.out.println("I am in methodI1 of class C1");
    }

    public void methodI2()
    {
        System.out.println("I am in methodI2 of class C1");
    }
}
```



```
-----
// Note that the class is declared as abstract as it does not
// satisfy the interface contract
abstract class D1 implements I2
{
    public void methodI1()
    {

    }

    //This class does not implement methodI2() hence declared abstract.
}

-----
public class InterFaceEx
{
    public static void main(String[] args)
    {

        I1 i1 = new B1();
        i1.methodI1(); // OK as methodI1 is present in B1
        //i1.methodI2(); Compilation error as methodI2 not present in I1
        //Casting to convert the type of the reference from type I1 to type I2

        ((I2) i1).methodI2();

        I2 i2 = new B1();

        i2.methodI1(); // OK

        i2.methodI2(); // OK

        //Does not Compile as methodA1() not present in interface reference I1
        //String var = i1.methodA1();
        //Hence I1 requires a cast to invoke methodA1

        String var2 = ((A1) i1).methodA1();

        System.out.println("var2 : " + var2);

        String var3 = ((B1) i1).methodA1();

        System.out.println("var3 : " + var3);

        String var4 = i1.toString();

        System.out.println("var4 : " + var4);

        String var5 = i2.toString();

        System.out.println("var5 : " + var5);

        I1 i3 = new C1();
    }
}
```

```
String var6 = i3.toString();

// It prints the Object toString() method
System.out.println("var6 : " + var6);

Object o1 = new B1();

// o1.methodI1(); does not compile as Object class does not define
// methodI1()
// To solve the problem we need to downcast o1 reference. We can do it
// in the following 4 ways

((I1) o1).methodI1(); // 1

((I2) o1).methodI1(); // 2

((B1) o1).methodI1(); // 3

/*
B1 does not have any relationship with C1 except they are "siblings".
Well, you can't cast siblings into one another.
*/

// ((C1)o1).methodI1(); Produces a ClassCastException

}
}
```

### Output

```
I am in methodI1 of class B1

I am in methodI2 of class B1

I am in methodI1 of class B1

I am in methodI2 of class B1

var2 : I am in methodA1 of class A1
var3 : I am in methodA1 of class A1
var4 : toString() method of class A1
var5 : toString() method of class A1
var6 : C1@190d11
I am in methodI1 of class B1
I am in methodI1 of class B1
I am in methodI1 of class B1
```

وراثة متعددة للواجهات

فئة يمكنها وراثه فئة واحدة أخرى فقط, لكن يمكنها أن ترث عدة واجهات (interface).  
الفئة الوارثة يجب عليها تطبيق كل الدوال – الطرق التي عُرِّفت داخل الواجهات.

```
public interface IPrint
{
    void Print();
}
```

```
public interface IMath
{
    int Sum();
    float Avg();
}
```

```
public class Sample implements IPrint1,IMath
{
```

```
    private int m_Num1;
    private int m_Num2;
    private int m_Num3;
```

```
    public Sample()
    {
    }
```

```
    public Sample(int n1,int n2,int n3)
    {
        m_Num1 = n1;
        m_Num2 = n2;
        m_Num3 = n3;
    }
```

```
    @Override public void Print()
    {
        System.out.println("num1="+ m_Num1 + " num2=" + m_Num2 +
            " num3=" + m_Num3);
    }
```

```
    @Override public int Sum()
    {
        return m_Num1+m_Num2+m_Num3;
    }
```

```
    @Override public float Avg()
    {
        return (float)Sum()/3;
    }
```

```
}
```

وراثه متعددة  
للواجهات

تطبيق لل- IPrint

تطبيق لل- IMath

تطبيق لل- IMath

```

class App
{
    public static void main(String[] args)
    {
        Sample s = new Sample(1,2,4);
        s.Print();
        System.out.println ("sum = " + s.Sum());
        System.out.println ("avg = " + s.Avg());
    }
}

```

كما ذكرنا الفئة الوارثة يجب عليها تطبيق كل الدوال – الطرق التي عُرِّفت داخل الواجهات , وأيضاً هنا إذا عرفنا موجه – مؤشر من احد الواجهات الموروثة يمكنه أن يؤثر على كائن من الفئات الوارثة (polymorphism)

مثال :

```

IPrint p = new Sample();
Imath m = new Sample();

```

بواسطة المؤشر p نستطيع أن نستدعي فقط الدالة Print الموجودة في الفئة Sample, بواسطة المؤشر m نستطيع أن نستدعي فقط الدالة Sum وأيضاً Avg. مبدأ تعدد الأشكال polymorphism, مؤشر من الفئة الأساسية Base يمكنه أن يؤثر على كائنات من الفئة الوارثة Derived صحيح هنا أيضاً في الواجهات.

سؤال :

ماذا يحدث إذا ورثت فئة معينة واجهتين مختلفتين , لكن الواجهتان تحويان عنوان متشابه لنفس الدالة؟  
 هنا يمكن كتابة دالة واحدة تخدم الواجهتين .

في هذه الحالة :

1. الدوال تكون من نوع `public` حتى وإن لم تكتب الكلمة .
2. تطبيق للدالة يجب أن يكون كدالة وهمية (virtual function) .

مثال :

```
public interface IPrint1
{
    void Print();
}
public interface IPrint2
{
    void Print();
}
```

```
class Sample:IPrint1,IPrint2
{
    private int m_Num;

    public Sample()
    {
    }
    public Sample(int num)
    {
        this.m_Num = num;
    }
    @Override public void Print()
    {
        System.out.println("num = " + m_Num);
    }
}
```

الفئة `Sample` ترث واجهتين , بداخلهم عناوين لدوال متشابهة .

تطبيق للدالة هذه الدالة تخدم كل الواجهات التي بها عنوان بإسم `Print`

```
class Application
{
    static void main(string[] args)
    {
        Sample s = new Sample(33);
        ((IPrint1)s).Print();
        ((IPrint2)s).Print();

        IPrint1 i1 = new Sample(12);
        i1.Print();

        IPrint2 i2 = new Sample(13);
        i2.Print();
    }
}
```

المناداة على الدالة تفرض علينا التحويل `casting`

موجه من نوع `IPrint1` يتعرف على الدوال الموجودة في `IPrint` .

```
num = 33
num = 33
num = 12
num = 13
```

المخرج للبرنامج :

مثال على وراثة واجهات لواجهات أخرى :

//Filename: Sports.java

public interface Sports

{

public void setHomeTeam(String name);

public void setVisitingTeam(String name);

}

//Filename: Football.java

public interface Football extends Sports

{

public void homeTeamScored(int points);

public void visitingTeamScored(int points);

public void endOfQuarter(int quarter);

}

//Filename: Hockey.java

public interface Hockey extends Sports

{

public void homeGoalScored();

public void visitingGoalScored();

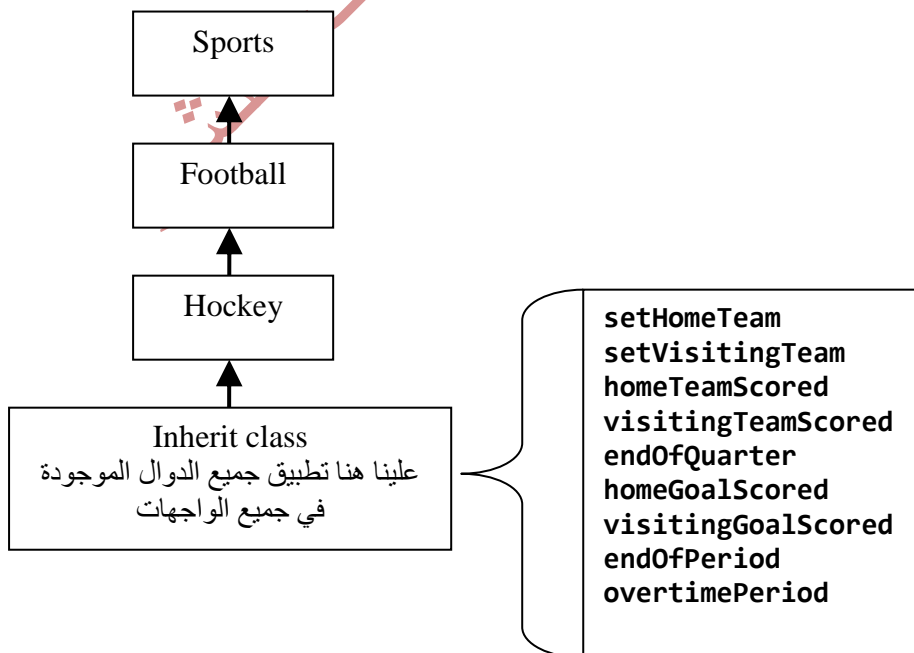
public void endOfPeriod(int period);

public void overtimePeriod(int ot);

}

رسم تخطيطي هرمي للواجهات :

واجهه ممكن أن ترث واجهة أخرى أو أكثر . الفئة التي ترث الواجهة الوارثة عليها تطبيق جميع الدوال التي بالواجهات كلها الموروثة والوارثة.



## تعرف على بعض الواجهات المعرّفة في لغة الجافا :

لغة الجافا تحوي بداخلها مجموعة من الواجهات يوجد أهمية كبيرة وجود هذه الواجهات التي تلزمنا بتطبيق بعض الدوال عند وراثتها .

### المثال الأول – الواجهة Comparable :

الواجهة تعرّف بداخلها الدالة

@Override public int compareTo(Object o)

هدف هذه الدالة أن تقوم بعمل مقارنة بين كائنين من نفس الفئة , أو من فئات مختلفة لكن لهم فئة اساسية مشتركة .

كل الفئات التي ترث هذه الواجهة تجبر على تطبيق الدالة compareTo

#### متى نستغل هذه الواجهة؟

عندما نريد أن نصنف كائنات من نفس النوع وحسب ميزة معينة .  
لغة الجافا تعرّف فئة من نوع List أو Arrays , وهي عبارة عن مجموعه من المؤشرات التي تؤشر على كائنات من نفس النوع .  
كل واحدة من هاتين الفئتين تحوي بداخلها دالة Sort , التي تقوم بتصنيف الكائنات حسب ميزة معينة Collections.sort() أو Arrays.sort()

```
public class Person implements Comparable
{
    private int person_id;
    private String name;

    /**
     * Compare current person with specified person
     * return zero if person_id for both person is same
     * return negative if current person_id is less than specified one
     *return positive if specified person_id is greater than specified one
     */
    @Override public int compareTo(Object o)
    {
        Person p = (Person) o;
        return this.person_id - o.person_id ;
    }
    .....
}
```

مثال :

```

public class Circle implements Comparable<Object>
{
    private int m_X;
    private int m_Y;
    private int m_R;
    //-----
    public Circle()
    {
    }
    public Circle(int m_X, int m_Y, int m_R)
    {
        this.m_X = m_X;
        this.m_Y = m_Y;
        this.m_R = m_R;
    }
    public Circle(Circle c1)
    {
        this.m_X = c1.m_X;
        this.m_Y = c1.m_Y;
        this.m_R = c1.m_R;
    }

    public void Print()
    {
        System.out.println("x =" + m_X + " y=" + m_Y);
        System.out.println("radius =" + m_R);
    }
    public double CalcArea()
    {
        double x = (Math.PI * (Math.pow(m_R,2)));
        return x;
    }
    public double CalcPerimeter()
    {
        return (double)(2 * Math.PI * m_R);
    }
    // Comparable implementation
    @Override public int compareTo(Object obj)
    {
        Circle c = null;
        if(obj instanceof Circle)
            c = (Circle)obj;
        else
            return 1;
        return this.m_R - c.m_R;
    }
}

```

تطبيق الدالة  
Comparable.CompareTo(Object)



```

class App
{
    static void main(String[] args)
    {
        Random rnd = new Random();
        Circle[] arr = new Circle[10];

        for(int i=0; i<10 ;i++)
        {
            int x1 = rnd.nextInt(200) ;
            int y1 = rnd.nextInt(200) ;
            int z1 = rnd.nextInt(200) ;
            arr[i] = new Circle( x1, y1, z1);
        }

        System.out.println("Unsorted array.");
        for(Circle o : arr)
        {
            System.out.println(o.CalcArea());
        }
        Arrays.sort(arr);

        System.out.println("Sorted Array.");
        for (int i=0; i< 10 ; i++)
        {
            if ( arr[i] instanceof Circle)
                System.out.println( ((Circle)arr[i]).CalcArea() ) ;
        }
    }
}

```

بناء مصفوفة من .Circles

طريقة 1 للطباعة  
لا حاجة للتحويل casting

طريقة 2 للطباعة

الفئة Circle ترث الواجهة Comparable , ولهذا يجب عليها تطبيق دالة المقارنة `Comparable.compareTo(Object obj)` وظيفه الدالة أن تقوم بمقارنة بين كائنين , بين `this` (الكائن الذي بواسطته فُعِلَت الدالة) وبين الكائن الذي تتلقاه الدالة (`obj`) .

هذا الكائن هو من نوع `Object` ولذلك يمكنه أن يكون من أي كائن معرف لدينا (يمكنه أن يكون من أي فئة نريد) لأن جميع الفئات ترث الفئة `Object` .

في البداية نفحص نوع الكائن الذي تتلقاه الدالة , يجب أن يكون من نوع Circle وبعدها نستطيع أن نقوم بالمقارنة.

```
if(obj instanceof Circle)
```

القيمة المعادة من الدالة-الطريقة تكون على النحو التالي :

- إذا كان الكائن this أكبر من الكائن obj - القيمة المعادة تكون عدد أكبر من 0 .
- إذا كان الكائن this مساوٍ للكائن obj - القيمة المعادة تكون 0 .
- إذا كان الكائن this أصغر من الكائن obj - القيمة المعادة تكون عدد أصغر من 0 .
- المقارنة تكون حسب معايير نحن نقررها (مثلا مساحة, نصف قطر, محيط الخ ..) .

الدالة Arrays.Sort(arr) تصنف مصفوفة الكائنات من نوع Circle. الدالة Arrays.Sort(arr) تتساعد (تستعمل) مع الدالة Circle.compareTo(Object) لكي تقوم بالمقارنة بين الكائنين من Circle. في هذه الدالة نقرر حسب أي معايير تتم عملية التصنيف وحسب أي نوع تصنيف تنازلي (Descending) أو تصاعدي (ascending).

الحل للمشكلة الأولى :

نعود الى السؤال الذي ذكر في بداية هذا الفصل , كيفية إجبار مجموعة من الفئات الموجودة داخل رسم تخطيطي هرمي (كالموجود في الصفحة السابقة) على تنفيذ دوال- طرق معينة , بينما لا نريد إجبار فئات أخرى التي هي أيضاً موجودة داخل نفس رسم التخطيط الهرمي على تنفيذ هذه الدوال ؟؟

الحل يكون بواسطة واجهات , نبني واجهة ترثها الثلاث فئات Derived11, Derived13 و Derived21 هكذا يعرفن دالة مشتركة لهن وتكون غير معرفة لباقي الفئات الأخرى .

## مثال 2 - الواجهات `Iterable<T>` , `Iterator<T>`

استعمال الامر `for` الذي يتحقق من الكائنات هي الطريقة الاسهل لتمشيط مصفوفة من الكائنات, بكلمات أخرى لا حاجة لعملية تحويل `casting`.

إذا أردنا تمشيط مصفوفة من الاعداد الصحيحة `int` المعرفة في الدالة الرئيسية - `main`, استعمالنا الامر `for` على النحو التالي :

```
int [] arr = new int [3];
for(int ob : arr)
{
    System.out.println( ob );
}
```

أيضاً إذا أردنا تمشيط مصفوفة من فئات مختلفة المعرفة في الدالة الرئيسية - `main`, نريد أن نستغل الامر `for` لتمشيط المصفوفة .

```
Person[] arr = new Person[10] ;
.
.
.
foreach(Person p : arr)
    p.Print();
```

لكن هل نستطيع استعمال `for` , في المثال التالي :

```
PersonArr arr = new PersonArr();
.
.
.
foreach(Person p : arr)
    p.Print();
```

الكائن `arr`, ليس مصفوفة انما مؤشر - موجه على فئة تحوي بداخلها مصفوفة كائنات من الفئة `Person`.

هنا إذا أردنا ان نستعمل الامر `for`, على الفئة أن ترث الواجهه `Iterable`, وعليها أن تنطبق الدالة `Iterator` .

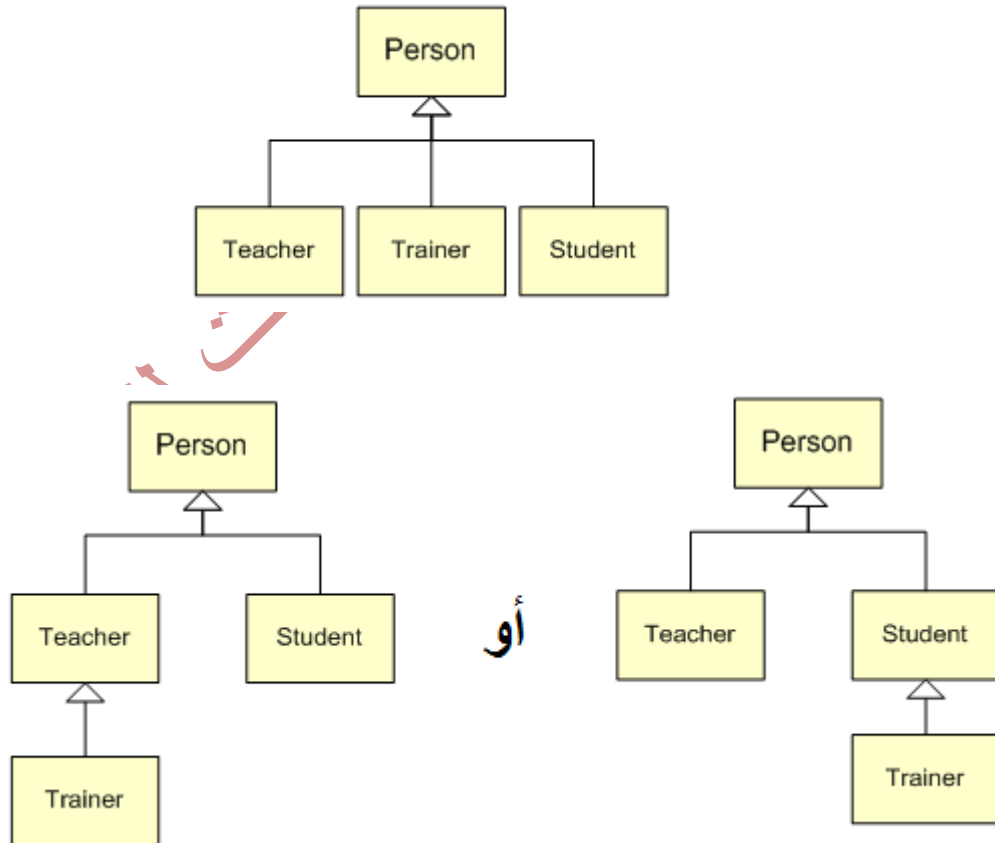
الحل للمشكلة الثانية:

حل هذه المشكلة معقد أكثر . لأنه يستعمل فئات وواجهات اضافية .

نعرف بعض الأساسيات :

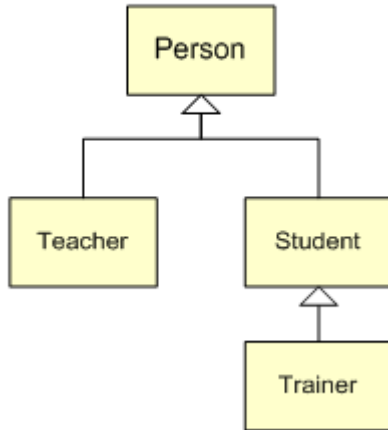
1. مساعد محاضر يجب ان يكون طالب جامعي.
2. المساعد يتلقى أجرته حسب ساعات العمل , المحاضر له أجره عامة. لكن اذا اشتغل المحاضر اكثر من 80 ساعة شهرية يأخذ زيادة على المعاش.
3. مساعد المحاضر (المعيد) لا يستطيع حساب علاماته, علامات المعيد مثل أي طالب جامعي وليس كمحاضر.

بأستعمال الواجهات يمكن أن نطبق الثلاثة إمكانيات للرسم التخطيطي الهرمي التالية بدون أن نكرر مقاطع كود :



الثلاثة إمكانيات للرسم التخطيطي الهرمي يمكن أن تكون مناسبة في حالات معينة .

في هذا المثال إختارنا أن نطبق الرسم التخطيطي الهرمي التالي :



السبب هو أنه هنالك عوامل مشتركة أكثر بين الطالب والمعيد (مساعد المحاضر) : مساعد يجب أن يكون دائماً طالب , الدوال المتعلقة بالعلامات هي نفسها بين الطالب وبين المساعد , ولكن بالنسبة لحساب الأجرة الشهرية هي ليست متشابهة بين المساعد والمحاضر.

هذا لا يعني أن الرسوم التخطيطية الهرمية الأخرى غير صحيحة , يمكن أن نطبقهم بنفس الدرجة من النجاعة .

في المرحلة الاولى هنالك دوال مشتركة بين بعض الفئات فنحددها :

- الدوال أو الطرق لحساب الاجرة الشهرية الصافية (بدون ضرائب) هي للمحاضر وهي نفسها لمساعد المحاضر .
- الدوال لإدارة العلامات , مثل معدل , عدد العلامات الفاشلة , طباعة تقارير الخ...

يمكن تطبيق الدوال المتعلقة بالعلامات في الفئة Student, وتقوم الفئة Trainer بوراثة هذه الفئة وكل الدوال بداخلها. المشكلة عدم تكرار الدوال المتعلقة بالأجرة الشهرية للمحاضر والمساعد,

لحساب الاجرة الشهرية:  
الاجرة الكلية - الضرائب = الاجرة الصافية

حتى نمتنع من عدم تكرار الدوال المتعلقة بالأجرة الشهرية للمحاضر والمساعد , نطبق هذه الدوال في فئة خاصة جديدة الموجودة في خارج التخطيط الهرمي . الحسابات للأجرة الشهرية تكون خارج الفئات Trainer و- Teacher .

نضيف فئة باسم Salary, هذه الفئة تحوي بداخلها الدوال المشتركة لحسابات الأجرة الشهرية بين المحاضر والمساعد . تقوم الفئة بحساب الأجرة للمحاضر وايضاً للمساعد على حد سواء.

يمكن أن يكون لكل منهما دالة لحساب الأجرة خاصة به.

هنا يأتي دور الواجهة نعرف واجهة جديدة أيضا نسميها ISalary, التي نعرف بداخلها الدوال اللازمة لحساب الأجرة لكل من المحاضر والمساعد , وكل من الفئات Teacher و- Trainer ترث هذه الواجهة .

الحل الكامل في الصفحات التالية ,

**المثال :**

```

public abstract class Person
{
    private String m_FirstName;
    private String m_LastName;
    private int m_ID;

    public Person()
    {
    }
    public Person(String first,String last,int id)
    {
        this.m_FirstName = first;
        this.m_LastName = last;
        this.m_ID = id;
    }
    public void Print()
    {
        System.out.println("Name:" + m_LastName + " " + m_FirstName);
        System.out.println("ID : "+ m_ID);
        PrintDetails();
    }
    protected abstract void PrintDetails();

    public String getFirstName()
    {
        return this.m_FirstName;
    }
    public String getLastName()
    {
        return this.m_LastName;
    }
    public int getID()
    {
        return this.m_ID;
    }
}

```

الفئة الأساسية المشتركة  
لجميع الفئات الأخرى

طباعة المعلومات , المعلومات المشتركة والمعلومات  
الخاصة لكل فئة مشتقة

```

public interface ISalary
{
    float CalcBruto();
}

```

الواجهة الجديدة التي  
بنيت لحساب الأجرة

```
public class Salary
{
    private Salary()
    {
    }
    public static float CalcSalary(ISalary obj)
    {
        float bruto = obj.CalcBruto();
        return(bruto-CalcIncomeTax(bruto)-CalcSocialInsuranceTax(bruto));
    }

    private static float CalcIncomeTax(float bruto)
    {
        return bruto * 0.3f;
    }
    private static float CalcSocialInsuranceTax(float bruto)
    {
        return bruto * 0.1f;
    }
}
```

الفئة التي تحسب الدوال العامة للأجرة الشهرية للمحاضر والمساعد

حساب الأجرة العامة بدون الضرائب

نسخة تجريبية - ليست للنشر



```

public class Student extends Person
{
    private int[] m_GradesArr = new int[10];
    private int m_GradesCounter;

    public Student()
    {
    }
    public Student(int id, String fname, String lname)
    {
        super(fname, lname, id);
    }

    public boolean AddGrade(int grade)
    {
        if (grade >= 0 && grade <= 100 &&
            m_GradesCounter < m_GradesArr.length)
        {
            m_GradesArr[m_GradesCounter++] = grade;
            return true;
        }
        return false;
    }
    public int CalcAvg()
    {
        int avg = 0;
        for (int i = 0; i < m_GradesCounter; i++)
            avg += m_GradesArr[i];
        return (int)(avg / m_GradesCounter);
    }

    public int FailuresNum()
    {
        int failures = 0;
        for (int i = 0; i < m_GradesCounter; i++)
            if (m_GradesArr[i] < 70)
                failures++;
        return failures;
    }
    protected @Override void PrintDetails()
    {
        System.out.println("Grades : ");
        for (int i = 0; i < m_GradesCounter; i++)
        {
            System.out.println( m_GradesArr[i]);
        }
        System.out.println();
        System.out.println("Average : " + CalcAvg());
        System.out.println("Failures : " + FailuresNum());
    }
}

```

الفئة للطالب، علامات  
حتى 10 علامات  
يعالج كل ما يتعلق  
بالطالب

```
public class Trainer extends Student implements ISalary
```

```
{
```

```
    private float m_SalPerHour;  
    private float m_Hours;
```

```
    public Trainer()
```

```
    {  
    }
```

```
    public Trainer(int id, String fn, String ln, float sal, float hours)
```

```
    {
```

```
        super(id, fn, ln);  
        this.m_SalPerHour = sal;  
        this.m_Hours = hours;
```

```
    }
```

```
    public void setSalaryPerHour(float sal)
```

```
    {
```

```
        this.m_SalPerHour= sal;
```

```
    }
```

```
    public float getSalaryPerHour()
```

```
    {
```

```
        return this.m_SalPerHour;
```

```
    }
```

```
    public void setHours(float h)
```

```
    {
```

```
        this.m_Hours= h;
```

```
    }
```

```
    public float getHours()
```

```
    {
```

```
        return this.m_Hours;
```

```
    }
```

```
    protected @Override void PrintDetails()
```

```
    {
```

```
        super.PrintDetails();  
        System.out.println("Salary: " + Salary.CalcSalary(this));
```

```
    }
```

```
    @Override public float CalcBruto()
```

```
    {
```

```
        return m_SalPerHour * m_Hours;
```

```
    }
```

```
}
```

ترث القدرة على ادارة العلامات من الفئة Student , تحسب الأجرة حسب الواجهه . ISalary

توجه الى - Salary  
لحساب المعاش الشهري.

```

public class Teacher extends Person implements ISalary
{
    private float m_BaseSalary;
    private float m_TeachingHours;
    public Teacher()
    {
    }
    public Teacher(int id, String fn,String ln, float sal, int hours)
    {
        super(fn, ln, id);
        this.m_BaseSalary = sal;
        this.m_TeachingHours = hours;
    }
    protected @Override void PrintDetails()
    {
        System.out.println("Base Salary :"+m_BaseSalary);
        System.out.println("Hours: "+m_TeachingHours);
        System.out.println("Salary:"+Salary.CalcSalary(this));
    }
    public void set_BaseSalary(float Bs)
    {
        this.m_BaseSalary= Bs;
    }
    public float get_BaseSalary()
    {
        return this.m_BaseSalary;
    }
    public void set_TeachingHours(float TH)
    {
        this.m_TeachingHours= TH;
    }
    public float get_TeachingHours()
    {
        return this.m_TeachingHours;
    }
    @Override public float CalcBruto()
    {
        float extra_payment = 0;

        if (m_TeachingHours > 80)
        {
            float extra_hours = m_TeachingHours - 80;
            float extra_payment_per_hour =m_BaseSalary/80*1.5f;
            extra_payment = extra_hours*extra_payment_per_hour;
        }
        return m_BaseSalary + extra_payment;
    }
}

```

توجه الى - Salary  
لحساب المعاش  
الشهري.

تطبيق ISalary, يتم بواسطة  
الاستدعاء للدالة  
. Salary.CalcSalary(...)

```
public class JavaApp
{
    /**
     * @param args the command line arguments
     */
    public static void main(String[] args)
    {
        Teacher t1 = new Teacher(1234,"Yosef","Ben Yazed" 2346.34f, 120);

        t1.Print();
        System.out.println();

        Trainer t2 = new Trainer(2345,"Rami","Ben Azez", 23.34f,87.5f);
        t2.AddGrade(100);
        t2.AddGrade(89);
        t2.AddGrade(98);
        t2.Print();

        System.out.println();
        Student s = new Student(4678,"Shadin","Ben Salih");
        s.AddGrade(76);
        s.AddGrade(85);
        s.AddGrade(89);
        s.Print();
    }
}
```

# أسئلة عن الواجهات

## سؤال 1 :

أ. جد الأخطاء بالبرنامج التالي وصححها. اشرح باختصار جوابك .

(سؤال بدون شرح لا يقبل)

```
public interface Person
```

```
{
```

```
    private void eat(int bananas);
```

```
    void drink(int coffee){ }
```

```
}
```

```
public class Employee extends Person {
```

```
    @Override public void eat(int bananas) {
```

```
        System.out.println ("I ate " + bananas + " bananas");
```

```
    }
```

```
}
```

```
public class Manager extends Employee
```

```
{
```

```
    @Override public new void eat(int bananas) {
```

```
        System.out.println ("I ate " + bananas * 2 + " bananas");
```

```
    }
```

```
    @Override public void drink(int coffee) {
```

```
        System.out.println ("I drank " + coffee + " cups of coffee");
```

```
    }
```

```
    @Override public void shout() {
```

```
        System.out.println ("I am the manager here");
```

```
    }
```

```
}
```

```
public static void main(String args[])
```

```
{
```

```
    Person p = new Employee();
```

```
    Employee e = new Manager();
```

```
    Manager m = (Manager)e;
```

```
    m.drink(10);
```

```
    p.eat(11);
```

```
    e.shout();
```

```
}
```

```
}
```

ماذا يكون المخرج بعد تصحيح الأخطاء؟

## الحل الأخطاء :

✓ الخطأ الأول في الواجهة person, كل العناوين من نوع public  
private void eat(int bananas);

✓ الخطأ الثاني لا يمكن كتابة تطبيق دالة داخل – interface  
void drink(int coffee){ }  
داخل الواجهة نضع عناوين الدوال بدون تطبيق .

✓ الخطأ الثالث داخل الفئة Employee يجب تطبيق الدالة drink  
@Override public void drink(int coffee)  
{  
System.out.println("I drank " + coffee + " cups of coffee");  
}

✓ الخطأ الرابع , داخل المناداة e.shout() على الدالة من الفئة الرئيسية غير ممكنة  
لأن المؤشر , لا يمكن التوجه المناداة على الدالة لأنه من نوع Employee  
وليس من نوع Manager . (للتصحيح نكتب m.shout()).

**سؤال 2:**

(أ) اقرأ المقطع التالي وأجب عن :

i. أرسم مخطط UML يصف بشكل صحيح ودقيق العلاقة بين الفئات .

ii. أضف للفئة Actor , عناوين الدوال – الطرق الناقصة .

<pre>public interface Dancer {     public void dance(); }  public interface Singer {     public void sing(); }  public interface Speaker{     public void speak(); }</pre>	<pre>public class Actor extends Dancer, Singer, Speaker {     private String name;     public Actor(String name) { ..}     // هنا ستظهر دوال أخرى ...     public void danceWith(Dancer dancer) {..} }</pre>
--	---

(أ) معطاة الفئة Test. اشرح لكل من الأسطر المعلمة ب (\*) اذا كانت صحيحة أم غير صحيحة وفسر جوابك .

```
public class Test
{
    public static void main (String[] args)
    {
        Actor actor1 = new Actor ("Rebekka");
        Actor actor2 = new Actor ("Mark");
        Dancer dancer1 = actor1;
        Dancer dancer2 = actor2;
        * dancer1.danceWith( dancer2);
        * Singer singer1 = new Singer();
        * Singer singer2 = new Actor ("Tom");
        * Dancer dancer3 = (Actor)singer2;
        * Actor actor3 = singer2;
        * Singer singer4 = (Singer) dancer3;
    }
} // end class Test
```

## الأخطاء:

✓ يجب تطبيق الدوال – الطرق الموجودة داخل الفئات الموروثة . يعني يجب تطبيق الدوال dance() , speak() , sing()

```
dancer1.danceWith( dancer2);
```

✓ هذا السطر خاطئ لا يوجد تعريف أو وصول الى الدالة الموجودة داخل  
class Actor

```
Singer singer1 = new Singer();
```

✓ هذا السطر خاطئ لا يمكن بناء كائن من الواجهات (interface)

```
Actor actor3 = singer2;
```

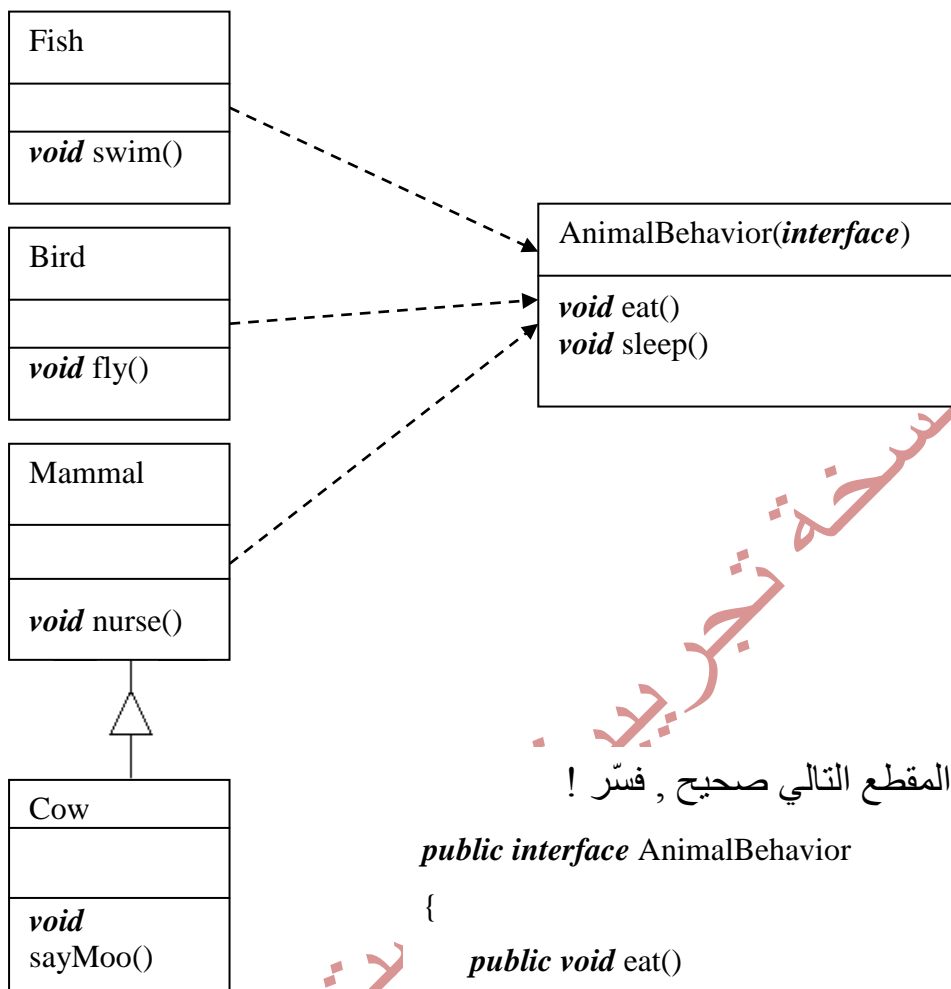
✓ هذا السطر خاطئ لا يوجد تحويل. يجب أن نقوم بعملية تحويل لكي نستطيع تنفيذ هذا الأمر .

نسخة غير رسمية - ليست للنشر



**سؤال 3:**

أنظر الى الرسم التخطيطي الذي أمامك وأجب عن الأسئلة التي تليه :



(أ) هل المقطع التالي صحيح , فسّر !

```

public interface AnimalBehavior
{
    public void eat()
    { System.out.println ( "I can eat"); }
    public void sleep()
    { System.out.println ( " I can sleep"); }
}
  
```

(ب) أكتب الفئات Bird و - Cow , وعناوين الدوال التي يجب أن تظهر داخل هذه الفئات .

(ت) (i) إشرح لكل من الأسطر المعلمة المرقمة اذا كانت صحيحة أم غير صحيحة وفسّر جوابك.

(ii) ما هي المشكلة الموجودة داخل الدالة ( ) `testAnimals()` في الفئة Zoo ؟ فسّر

```
public static void main(String[ ] args)
{
    AnimalBehavior[ ] animals = new AnimalBehavior[4]; .....1
    animals[0] = new Mammal(); .....2
    animals[1] = new AnimalBehavior (); .....3
    animals[2] = new Cow(); .....4
    animals[3] = new Fish();

    for(int i=0 ; i < 4; i++)
    {
        animals[i].sleep(); .....5
    }
    animals[2].sayMoo(); .....6
}
```

```
public class Zoo
{
    public static void testCow(Cow cow)
    {
        cow.nurse().....7
    }

    public static void testFish(AnimalBehavior animal)
    {
        animal.swim();.....8
    }
    public static void testAnimals(AnimalBehavior animal)
    {
        (Bird)animal.fly();.....9
        (Fish)animal.swim();.....10
    }
}
```

### حل للسؤال 3 :

#### حل قسم أ :

الحل غير صحيح , لا يمكن تطبيق دوال – طرق داخل الواجهات . نعلم أن الفئة AnimalBehavior هي واجهة ولهذا لا يمكن أن نطبق نكتب داخلها فئات فقط عناوين.

#### حل قسم ب :

داخل الفئة fish , يجب أن يكون تطبيق للدوال ( sleep() , eat() , وأيضاً الدالة swim() .  
نفس الشيء للفئة Bird .

```
class fish implements AnimalBehavior
{
    @Override public void eat()
    {
        System.out.println ("class Fish .. eat");
    }
    @Override public void sleep()
    {
        System.out.println ("class Fish .. sleep");
    }
    public void swim()
    {
        System.out.println ("class Fish .. Swim");
    }
}

class Bird implements AnimalBehavior
{
    @Override public void eat ()
    {
        System.out.println ("class Bird .. eat");
    }
    @Override public void sleep ()
    {
        System.out.println ("class Bird .. sleep");
    }
    public void fly ()
    {
        System.out.println ("class Bird .. fly");
    }
}
```

#### حل قسم ج :

1. السطر 1 صحيح , بناء مصفوفة من نوع الفئة الأساسية (من نوع الواجهة) .

2. السطر 2 , صحيح الخلية الاولى (الموجه الاول) يؤشر على كائن من نوع

Mamal. اذا كان لدينا فئة تراث الواجهه فإن حسب مفهوم تعدد الأشكال

(polymorphism) موجه من نوع فئة الأساس (AnimalBehavior) يمكنه أن

يؤشر - يوجه على كائنات من نوع الفئات المشتقة (Mamal).

3. السطر 3 , غير صحيح لا يمكن بناء كائنات من الموجهات .

4. السطر 4 , صحيح موجه – مؤشر من نوع الفئة الأساسية يمكنه أن يوجه –  
يؤشر على كائنات من الفئات المشتقة .

5. السطر 5 , صحيح بشرط أن نكون قد صححنا الخطأ الموجود في السطر 3 ,  
لأنه عندها كل الكائنات لها الدالة – الطريقة sleep . وراثه واجهة تجبر على  
تطبيق الدوال بداخلها .

6. السطر 6 , غير صحيح يجب أن نتأكد بأن الكائن من نوع Cow . وبعدها نحول  
الموجه – المؤشر الى نوع الفئة Cow .

```
if (animals[2] instanceof Cow)
{
    Cow c1 = (Cow) animal[2]; } // ( (Cow)animal[2] ).sayMoo( );
    c1.sayMoo();
}
```

7. السطر 7 , صحيح البارامتر (Cow cow) الذي تتلقاه الدالة من نوع Cow . لذلك  
يمكنه أن ينادي – يستدعي الدالة nurse الموجودة في الفئة Mammal بسبب  
الوراثه.

8. السطر 8 , غير صحيح يجب التأكد من أن الموجه – المؤشر animal بالفعل يؤشر  
على كائن من نوع الفئة fish .  
مشابه للسطر 6 .

```
if ( animal instanceof fish)
((fish)animal).swim();
```

9. الأسطر 9 و 10 , غير صحيحة يجب التأكد من نوع الكائن قبل إستدعاء الدالة .

```
if( animal instanceof Bird )
    ((Bird)animal).fly();

if (animal instanceof fish )
    ((fish)animal).swim();
```

**سؤال 4 :**

في إحدى المؤسسات مجموعتين من الطلاب A و -B , مجموعة الصغار تكون من الفئة AStudent , ومجموعة الكبار الفئة BStudent . المجموعتان تَرثان الواجهة Student التي تظهر بالاسفل لكل طالب يوجد اسم ورقم هوية.

لكل مجموعة يوجد مُرَكِّز لهذه المجموعة . وهو معلم يُمَثِّل بواسطة الفئة Teacher .  
العلامات في المؤسسة تكون حسب المجموعة وليست فردية , ومركز المجموعة يمكنه زيادة أو نقصان لعلامات المجموعة .

يمكن لطالب من مجموعة الصغار أن يأخذ تأشيرة للخروج من المؤسسة اذا كانت علامات مجموعته أكبر من 150 , ولديه إذن من مركز المجموعة .  
يمكن لطالب من مجموعة الكبار أن يأخذ تأشيرة للخروج من المؤسسة اذا كانت علامات مجموعته أكبر من 200 .

معطى المقطع التالي :

```
public interface Student
{
    /* دالة تعدّل علامات المجموعة حسب متغير grade */
    public void SetGroupGrade (int grade);

    /* دالة تعيد علامات المجموعة */
    public int GetGroupGrade();

    /* دالة تعيد مركز المجموعة بشكل كائن */
    public Teacher GetGroupTeacher();

    /* دالة تغيّر مركز المجموعة */
    public void SetGroupTeacher (Teacher newTeacher);

    /* دالة تعلن عن حصول الطالب على إذن أم لا */
    public bool HasPermission();
}
```

أ. إرسم مخطط UML الذي يصف العلاقة بين الفئات كيفما ذكرت بالسؤال .

ب. أمامك تطبيق للدالة `bool HasPermission()` الموجودة داخل الفئة `AStudent` :

```
public bool HasPermission()
{
    if ((AStudent.grade > 150) && (AStudent.GroupTeacher.GivePermission (this)))
        return true;
    else
        return false;
}
```

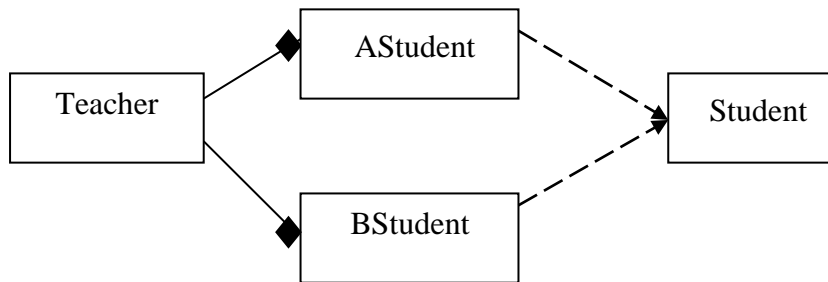
أكتب عنوان الدالة `(...) GivePermission` التي تظهر بالفئة `Teacher`.

ت. أكتب الفئة `AStudent` على النحو التالي الميزات وعناوين الدوال التي يمكن أن تكون داخل الفئة (لا حاجة لكتابة الدالة بشكل كامل فقط العناوين) .

ث. في البرنامج الرئيسي عُرِّفَت مصفوفة التي تحوي طلاب من المجموعتين داخل المؤسسة . البرنامج يفحص الطلاب الذين لهم تأشيرة خروج من المؤسسة . أي مفهوم من مفاهيم برمجة الكائنات الموجهة يُفَعَّل عند عملية الفحص . اشرح .

**حل سؤال 4 :**

أ. رسم تخطيطي



ب. عنوان الدالة :

```
public bool givePermission (Student std)
```

ويمكن أن تكون بهذا الشكل أيضا:

```
public bool givePermission (AStudent std)
```

هذه الدالة خاصة فقط للطلاب الصغار .

ت. هنا نكتب الحل الكامل فقط لمن يريد برمجة السؤال بشكل كامل.

```
public class AStudent implements Student
{
```

```
    private static int grade;
```

```
    private static Teacher groupTeacher;
```

```
    private String name;
```

```
    private int idNumber;
```

```
    public void setGroupGrade (int grade)
```

```
    {
```

```
        Astudent.grade = grade;
```

```
    }
```

```
    public int getGroupGrade()
```

```
    {
```

```
        return AStudent.grade;
```

```
    }
```

```
    public Teacher getGroupTeacher()
```

```
    {
```

```
        return AStudent.groupTeacher;
```

```
    }
```

```
    public Teacher setGroupTeacher (Teacher newTeacher)
```

```
    {
```

```
        AStudent.groupTeacher = newTeacher; // لا نبني كائن منسوخ
```

```
    }
```

```

public bool hasPermission()
{
    if ((AStudent.grade > 150) && (AStudent.groupTeacher.givePermission (this)))
        return true;
    else
        return false;
}
}

```

عند فحص الكائنات نستدعي الدالة hasPermission() لكل واحد من الكائنات الموجودة داخل المصفوفة , لكل طالب نستدعي نفعل دالة لها نفس الاسم التي تقوم بأشياء مختلفة حسب المجموعة التي ينتمي إليها . هذا المفهوم هو مفهوم تعدد الأشكال حسب نوع الكائن نستدعي الدالة - الطريقة الملائمة .

### ملاحظات:

- أ. في القسم ب, يجب أن نفهم ان الدالة تعيد قيمة منطقية (True/False) حسب نوع الكائن والدالة تتلقى طالب من المجموعتين .
- ب. في القسم ت , يجب التركيز على أن تكون الميزات ستاتية (static) .
- ت. في القسم ت , الدوال الاجبارية هي المعرفة التي تظهر بالواجهه Student. لا حاجة لتطبيق الدوال .
- ث. في القسم ج , يجب الانتباه لمفهوم تعدد الاشكال .



**سؤال 5:**

أ. معطى البرنامج التالي افرض أن جميع الأسطر صحيحة

```
public static void main(String[ ] args)
{
    B b1 = new A();
    B b2 = new C();
    A a1 = (C) b2;
    D d1 = new C();
    B b1 = new B();
    D d2 = new A();
}
```

- (1) ارسم مخططي UML, يصفان العلاقات الممكنة بين الفئات التي عُرِّفت في البرنامج الرئيسي.
- (2) افرض أن السطر الأخير في البرنامج يقوم بعملية تحويل غير صحيحة . ارسم مخطط UML جديد يصف العلاقات الممكنة بين الفئات التي عُرِّفت في البرنامج الرئيسي.

ب. أمامك مقطع صحيح :

```
public class What
{
    public static void main (String[] args)
    {
        A a = new A ( );
        A m = new A ( );
        m.SetColor ("Red");
        System.out.println (m.toString ( ));
        *** a.SetColor ("Blue");
        System.out.println (m.toString ( ));
    }
}
```

1. تشغيل البرنامج يظهر المخرج التالي :

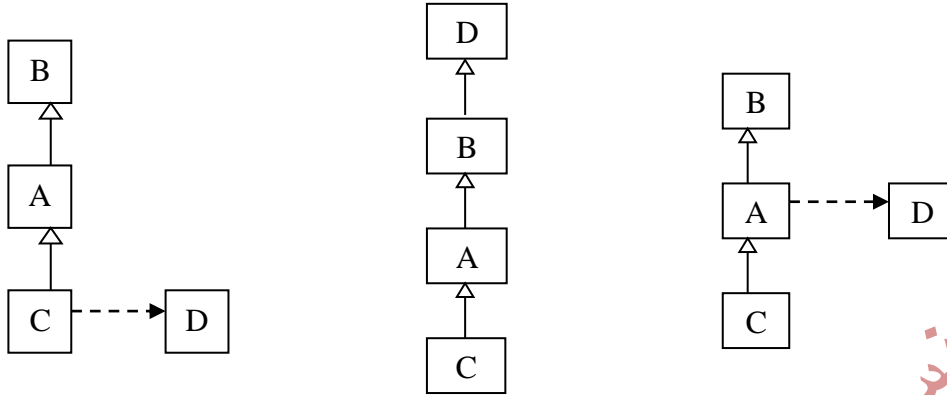
A Red object  
A Blue object

ما هي الميزة التي يجب أن تُعرّف داخل الفئة A , اشرح .

2. أكتب السطر \*\*\* بصورة اخرى , اشرح ماذا فعلت .

حل سؤال 5 :

أ.



هناك عدة احتمالات للحل :

1. الميزة التي يجب أن تظهر هي:

protected static String color;

واضح أن هناك ميزة لون في الفئة A, لأنه يوجد دالة التي تعدّل وتغير الميزة .  
 من الكود واضح أن تغير الميزة يكون عن طريق مؤشر a, وأيضا عن طريق مؤشر  
 آخر m, ولهذا نفهم أن الميزة تُعرّف كميزة ستاتية static.  
 الميزة الستاتية تكون تابعة لكل الكائنات وهي لا تكون داخل الكائن , انما ميزة خارجية  
 تابعة لكل الكائنات تخدم كل الكائنات . كل الكائنات تستطيع الوصول الى هذه الميزة  
 وتعديل قيمتها أو تأخذ هذه القيمة وكل تغيير لقيمة هذه الميزة يكون معروف لكل الكائنات  
 الاخرى.

2. يمكن كتابة السطر بصورة : A.setColor ("Blue");

لأن الميزة هي ستاتية لهذا يمكن تغير قيمة الميزة بصورة مباشرة من داخل الفئة .

أو يمكن  
 عن طريق مؤشر (كائن) آخر يمكن تعديل القيمة .  
 m.setColor ("Blue");

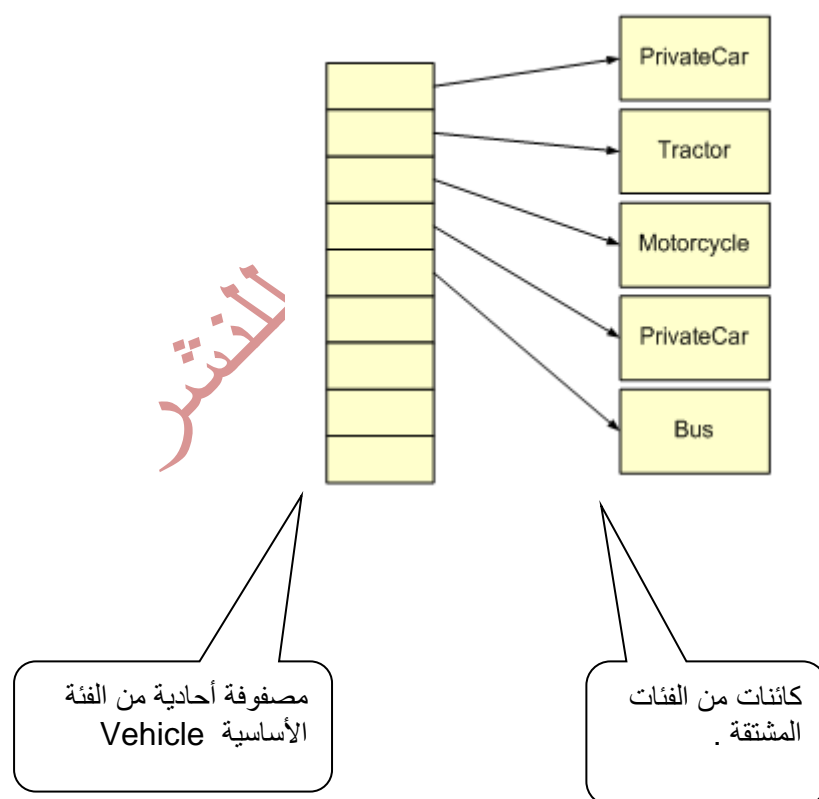
# ArrayList, Collections And Polymorphism

## ArrayList, Collections And Polymorphism

عادةً الوزارات العامة في أي دولة تقوم بحفظ كميات هائلة من المعطيات , عن الأشخاص والعائلات , عن المنازل وعناوين المنازل , عن الطلاب واسماء الطلاب , عن المدارس والمدرسين, الخ..

إذا أخذنا وزارة المواصلات فإنها تحفظ معطيات ضخمة عن كل السيارات في الدولة , سيارات خاصة, سيارات عامة , سيارات أجرة بأنواعها, شاحنات , باصات , الخ ... يمكن أن نقول بالتأكيد أن كل الفئات ترث الفئة الأساسية المشتركة لكل السيارات ونفرض أننا أعطيناها الاسم `Vehicle`.

مثال :



مثال على كيفية عمل البرنامج:

```
abstract class Vehicle
{
    public abstract void Print();
}

-----
abstract class Car extends Vehicle
{
    public @Override void Print()
    {
        System.out.println("Car class");
    }
}

-----
class Motorcycle extends Vehicle
{
    public @Override void Print()
    {
        System.out.println ("Motorcycle class");
    }
}

-----
class Track extends Car
{
    public @Override void Print()
    {
        System.out.println ("Track class");
    }
}

-----
class Semitrailer extends Car
{
    public @Override void Print()
    {
        System.out.println ("Semitrailer class");
    }
}

-----
class Bus extends Car
{
    public @Override void Print()
    {
        System.out.println ("Bus class");
    }
}
```

فئات مشتقة

```
class PrivateCar extends Car
{
    public @Override void Print()
    {
        System.out.println ("PrivateCar class");
    }
}
```

```
-----
class Van extends Car
{
    public @Override void Print()
    {
        System.out.println("Van class");
    }
}
```

```
-----
class VehiclesArr
{
    private Vehicle[] m_Arr;
    private int m_Counter;
    public VehiclesArr(int size)
    {
        m_Arr = new Vehicle[size];
    }
    public void AddCar(Vehicle v)
    {
        if (m_Counter < m_Arr.Length)
            m_Arr[m_Counter++] = v;
    }
    public void Print()
    {
        for (Vehicle v : m_Arr)
            v.Print();
    }
}
```

إضافة كائن من الفئات  
المشتقة الى المصفوفة.

طباعة المصفوفة

```
-----
class App
{
    static void main(String[] args)
    {
        VehiclesArr Arr = new VehiclesArr(6);
        Arr.AddCar(new Bus());
        Arr.AddCar(new PrivateCar());
        Arr.AddCar(new Track());
        Arr.AddCar(new Semitrailer());
        Arr.AddCar(new Motorcycle());
        Arr.AddCar(new PrivateCar());
        Arr.Print();
    }
}
```

إضافة كائنات الى  
المصفوفة

في لغة الجافا وكما في لغة ال NET، المصفوفة الأحادية هو مبنى معطيات له حجم ثابت عادة نحدده عند تعريف المصفوفة .

حجم (عدد العناصر) تحدّد عند التعريف، والوصول إلى محتواه يكون بواسطة المؤشرات. مبنى المعطيات هذا له حسنات عدة , منها الوصول إلى مكان في المصفوفة هو سريع جدا.

ومع ذلك، ليس دائماً نريد أن يكون كبر المصفوفة ثابت وغالبا كبر المصفوفة لا يكون معروف من قبل , إنما يكون في وقت التشغيل (Run Time) .

عندما يكون حجم المصفوفة غير معروف ، فإن الافد الحاجة ضلية لإستعمال مبانٍ جاهزة موجودة في لغة الجافا تمكننا من تغيير حدود أو حجم مبنى المعطيات عن `java.util.AbstractCollection`.

## الفئة ArrayList

فئة ArrayList إمتداد للفئة AbstractList وتطبق الواجهة List. ArrayList تدعم المصفوفات الدينامية (المتغيرة ليست الثابتة) التي يمكن أن تكبر حسب الحاجة. في لغة الجافا، المصفوفات الأحادية ذات طول ثابت.

بعد أن يتم إنشاء مصفوفة عادية ، فإنها لا يمكن أن تكبر أو تصغر، وهو ما يعني أنه يجب أن نعرف مسبقا كم عدد عناصر المصفوفة. ولكن، في بعض الأحيان، قد لا نعرف حتى وقت التشغيل بالضبط كبير المجموعة التي تحتاج إليها. للتعامل مع هذه الحالة، يحدد إطار مجموعات ArrayList. في جوهرها، على ArrayList هو مجموعة متغيرة الطول من المؤشرات على الكائنات. وهذا هو، على ArrayList يمكن زيادة أو نقصان حيوي-دينامي في الحجم. يتم إنشاء قوائم مجموعة مع الحجم الأولي. عندما يتم تجاوز هذا الحجم، يتم تكبير تلقائيا المجموعة. عندما تتم إزالة الكائنات، قد تنقلص المصفوف.

ثلاث بناءات للفئة :

ArrayList( )

ArrayList(Collection c)

ArrayList(int capacity)

البناء 1 يبني مجموعة فارغة البناء الثاني يتلقى مجموعة ويبني خلاياه من هذه المجموعة . البناء 3 يبني المجموعه حسب قدرة استيعاب أولية, هذا الكبر هو حجم المصفوفة الأساسية التي تستخدم لتخزين العناصر. القدرة- الكبر تكبر تلقائيا كلما تم إضافة عناصر إلى المجموعة .



البرنامج التالي يوضح استخدام بسيط من ArrayList. يتم إنشاء مجموعة، ثم يتم إضافة كائنات من نوع String ثم يتم عرض القائمة. تتم إزالة بعض العناصر ويتم عرض المجموعة مرة أخرى.

### مثال :

```
public static void main(String[] args)
{
    // create an array list
    ArrayList al = new ArrayList();
    System.out.println("Initial size of al: " + al.size());
    // add elements to the array list

    al.add("C");
    al.add("A");
    al.add("E");
    al.add("B");
    al.add("D");
    al.add("F");
    al.add(1, "A2");

    System.out.println("Size of al after additions:" + al.size());
    // display the array list
    System.out.println("Contents of al: " + al);
    // Remove elements from the array list
    al.remove("F");
    al.remove(2);
    System.out.println("Size of al after deletions:" + al.size());
    System.out.println("Contents of al: " + al);
}
```

المخرج :

Initial size of al: 0  
Size of al after additions: 7  
Contents of al: [C, A2, A, E, B, D, F]  
Size of al after deletions: 5  
Contents of al: [C, A2, E, B, D]

نلاحظ أن المجموعة A1 تبدأ فارغة من ثم تكبر كلما تم إضافة عناصر إليها. عندما تتم إزالة عناصر يتم تقليص حجمها.

على الرغم من أن قدرة الكائن ArrayList أن يزداد تلقائياً كلما تم تخزين كائنات جديدة في المجموعة ، لكن يمكنك أيضاً زيادة قدرة-كبر الكائن ArrayList يدوياً عن طريق `ensureCapacity()`.

`void ensureCapacity(int cap)`

إذا كنت ترغب في تقليل حجم المصفوفة التي تركز عليها ArrayList بحيث يكون بالضبط كبرها كعدد العناصر أو الكائنات الفعلية الموجودة في المجموعة . `trimToSize()`

الدوال والميزات الموجودة داخل الفئة ArrayList:

إضافي بمكان معين	<b><code>void add(int index, Object element)</code></b> Inserts the specified element at the specified position index in this list. Throws <code>IndexOutOfBoundsException</code> if the specified index is out of range ( <code>index &lt; 0    index &gt; size()</code> ).
إضافة كائن	<b><code>boolean add(Object o)</code></b> Appends the specified element to the end of this list.
إضافة مجموعة معينة	<b><code>boolean addAll(Collection c)</code></b> Appends all of the elements in the specified collection to the end of this list, in the order that they are returned by the specified collection's iterator. Throws <code>NullPointerException</code> if the specified collection is null.

محي كل مكونات	<b>void clear()</b>  Removes all of the elements from this list.
نسخ بشكل كامل	<b>Object clone()</b>  Returns a shallow copy of this ArrayList.
هل الكائن موجود داخل	<b>boolean contains(Object o)</b>  Returns true if this list contains the specified element. More formally, returns true if and only if this list contains at least one element e such that (o==null ? e==null : o.equals(e)).
زيادة حجم ال ArrayList	<b>void ensureCapacity(int minCapacity)</b>  Increases the capacity of this ArrayList instance, if necessary, to ensure that it can hold at least the number of elements specified by the minimum capacity argument.
إرجاع كائن بخلية معينة	<b>Object get(int index)</b>  Returns the element at the specified position in this list. Throws <code>IndexOutOfBoundsException</code> if the specified index is out of range ( <code>index &lt; 0    index &gt;= size()</code> ).
إعادة مكان لكائن معين في أول مكان يجده	<b>int indexOf(Object o)</b>  Returns the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.
إعادة مكان لكائن معين في آخر مكان يجده	<b>int lastIndexOf(Object o)</b>  Returns the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.
محي كائن في مكان محدد وإعادته	<b>Object remove(int index)</b>  Removes the element at the specified position in this list. Throws <code>IndexOutOfBoundsException</code> if index out of range ( <code>index &lt; 0    index &gt;= size()</code> ).
محي مجموعة من الكائنات من حد ال حد آخر	<b>protected void removeRange(int fromIndex, int toIndex)</b>  Removes from this List all of the elements whose index is between fromIndex, inclusive and toIndex, exclusive.

وضع في مكان	<p><code>Object set(int index, Object element)</code></p> <p>Replaces the element at the specified position in this list with the specified element. Throws <code>IndexOutOfBoundsException</code> if the specified index is out of range (<code>index &lt; 0    index &gt;= size()</code>).</p>
حجم ال ArrayList	<p><code>int size()</code></p> <p>Returns the number of elements in this list.</p>
تحويل الى مصفوفة من نوع Object	<p><code>Object[] toArray()</code></p> <p>Returns an array containing all of the elements in this list in the correct order. Throws <code>NullPointerException</code> if the specified array is null.</p>
	<p><code>Object[] toArray(Object[] a)</code></p> <p>Returns an array containing all of the elements in this list in the correct order; the runtime type of the returned array is that of the specified array.</p>
تحديد حجم	<p><code>void trimToSize()</code></p> <p>Trims the capacity of this <code>ArrayList</code> instance to be the list's current size.</p>

# أسئلة متنوعة وحلول

من المهم جداً تطبيق الأسئلة على الحاسوب لفهم الموضوع بشكل أفضل .  
أسئلة عامة من الإنترنت ومن امتحانات مدرسية وجامعية .

**سؤال 1 :**

أمامك المقطع التالي :

```
public class Stam
{
    public void B(int m, int n)
    {
        System.out.println(m * n);
    }

    public void f()
    {
        System.out.println ("inside Stam");
    }
}

public class B extends Stam
{
    protected int y;
    public B(int x)
    {
        this.y = x;
    }
}

public class C extends Stam
{
    public int x;
    public static void g()
    {
        System.out.println("inside C");
    }
}

public class Test
{
    public static void main(String[] args)
    {
        C temp = new C();
    }
}
```

أجب عن البنود "أ" – "د" ، متطرقاً إلى البرنامج المعطى .

أ .

هل يمكن تغيير قيمة الصفة x المعرفة في الفئة C ، من داخل طريقة في الفئة Stam؟  
إذا كانت الإجابة نعم - اكتب كيف، إذا كانت الإجابة لا - علّل لماذا .

ب .

(i) اكتب أمراً أو أوامر في الطريقة main ، لاستدعاء الطريقة B المعرفة في الفئة Stam.  
(ii) اكتب أمراً أو أوامر في الطريقة main ، لاستدعاء البناء B المعروف في الفئة B.

ج .

أمامك ثلاثة أوامر i - iii .  
بالنسبة لكل واحد منها، حدد إذا كان يمكن إضافته في الطريقة main .  
علّل تحديداتك بواسطة مميزات برمجة موجهة كائنات .  
i Stam obj = new B(7);  
ii Stam obj = new B ();  
iii B obj = new Stam();

د .

هل يمكن استدعاء الطريقة f() المعرفة في الفئة Stam، من داخل طريقة في الفئة B ؟ إذا كانت الإجابة نعم - اكتب الاستدعاء ، إذا كانت الإجابة لا - علّل لماذا .

هـ .

هل يمكن استدعاء الطريقة g() المعرفة في الفئة C، من داخل طريقة في الفئة main ؟ إذا كانت الإجابة نعم - اكتب الاستدعاء، إذا كانت الإجابة لا - علّل لماذا .

**حل سؤال 1:**

(أ) الجواب هو لا , لا يمكن لدالة أو طريقة معينة داخل الفئة Stam أن تتوجه الى كائن موجود داخل الفئة C. دالة بداخل base لا يُمكنها أن تغيّر قيمة ميزة داخل فئة Derived حتى وإن كانت الميزة معرفة كـ public.

إذا عرّفنا كائن من نوع داخل الفئة Stam يمكنه أن يغيّر قيمة الميزة x مباشرة لأنها من نوع public.

(ب) كتابة مقطع داخل Main –

```
Stam s1 = new Stam();
s1.B(5, 5); // استدعاء الدالة B الموجودة داخل Stam
             // هذه دالة داخلية في الفئة Stam

B b1 = new B(4); // استدعاء الدالة B الموجودة داخل B
                 // هذه دالة داخلية في الفئة B
```

(ت)

- (i) صحيحة , مؤشر من base يُؤشر على كائن من نوع Derived.
- (ii) غير صحيحة , يجب إرسال قيمة واحدة للبناء B .
- (iii) غير صحيحة , مؤشر من Derived لا يمكنه أن يُؤشر على كائن من نوع base بدون تحويل.

(ث) الجواب نعم , نكتب هذه الفئة داخل الفئة B .

```
public void Call_f()
{
    this.f();
}
```

(ج) الجواب هو نعم , نكتب الإستدعاء للدالة في ال Main – بالصورة التالية

```
C.g();
```

السبب ان الدالة ستاتيية (ساكنة) لا حاجة لبناء كائن لإستدعاء الدالة .  
يمكن استدعاء الدالة بشكل مباشر .



**سؤال 2 :**

معطاة الفئات الثلاث : NameA , NameB , NameC والفئة الرئيسية Test .  
تتبع بواسطة جدول متابعة تنفيذ الطريقة main في الفئة Test , وأكتب المخرج .  
على الجدول أن يشمل قيم المتغيرات ، وبالنسبة لكل كائن - قيم صفاته

```
public class NameA
{
    protected int x;

    public NameA(int x)
    { this.x = x; }

    public int getX()
    { return x; }

    @Override public String toString()
    { return "NameA: x=" + x; }

    public String display()
    { return "display NameA"; }
}

//-----

public class NameB extends NameA
{
    protected int y;

    public NameB(int x, int y)
    {
        super(x);
        this.y = y;
    }

    @Override public String toString()
    { return "NameB: x=" + getX() + "y= " + this.y; }
}

//-----

public class NameC extends NameB
{
    protected int z;
    public NameC(int x, int y, int z)
    {
        super(x, y);
        this.z = z;
    }
}
```

```
}

@Override public String toString()
{ return " NameC:  x=" + x + "y=" + y + "z=" + z; }

@Override public String display()
{ return "display NameC"; }
}

//-----
public class Test
{
    public static void main(String[] args)
    {
        NameC nc1 = new NameC(1, 2, 7);
        System.out.println(nc1.toString());

        NameA people[] = new NameA[4];

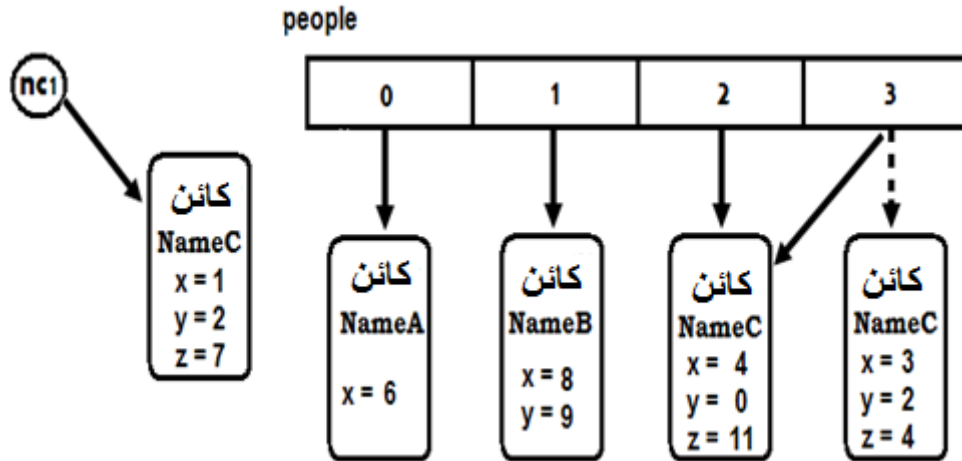
        people[0] = new NameA(6);
        people[1] = new NameB(8, 9);
        people[2] = new NameC(4, 0, 11);
        people[3] = new NameC(3, 2, 4);

        people[3] = (NameA)people[2];
        int count = 0;
        for (int i = 0; i < people.length; i++)
        {
            System.out.println(people[i].display());
            if (people[i].getX() > 5)
            {
                count++;
            }
        }
        System.out.println(count);
        count = 0;

        for (int i = 0; i < people.length; i++)
        {
            if (people[i] instanceof NameC)
            {
                count++;
            }
        }
        System.out.println(count);
    }
}
```

**حل سؤال 2:**

مخطط يصف وضعية الكائنات والمؤشرات في البرنامج :



المصفوفة معرفّة من نوع الفئة الاساسية NameA, المخرج للبرنامج يكون بالصورة التالية .

print NameC: x=1 y=2 z=7

طباعة الكائن nc1

display NameA

طباعة الكائن الاول في المصفوفة , استعمال الدالة display الخاصة به.

display NameA

الكائن من نوع NameB, لا يوجد له دالة display لكنه استعمل الدالة التي ورثها والموجودة داخل NameA .

display NameC

display NameC

المؤشران يؤشران على نفس الكائن , بسبب أن الدالة display هي وهمية (virtual) داخل NameA لهذا نُقِّدَت الدالة الموجودة في NameC.

2

عدّ عن طريق الحلقة الاولى كم من الكائنات قيمة الـ x لهم أكبر من 5 .

2

عدّ عن طريق الحلقة الثانية كم من الكائنات من نوع NameC يوجد .

**سؤال 3 :**

قررت شركة سيارات أجرة حوسبة إدارة سياراتها . لهذا الغرض عرفوا الفئة سيارة أجرة (Taxi) ، التي لها الصفات التالية :

- رقم الترخيص (int taxiId)
- اسم السائق (String driverName)
- عدد مقاعد الركاب (عدا السائق) int numPass
- هل سيارة الأجرة غير مشغولة – (Boolean available)

الصفة available مبدئية لـ true.

كذلك عُرِّفت عمليات الإعادة :

getId() , getName() , getNumPass() , isAvailable()

أ. طبق بلغة جافا في الفئة Taxi الطرق البنائية التالية :

- طريقة تتلقى كبار امترين اسم السائق ورقم الترخيص، وتبتدئ بالتلاؤم driverName و taxiId ، وتبتدئ numPass ليكون 4 .
- طريقة تتلقى كبار امترات اسم السائق ورقم الترخيص وعدد مقاعد الركاب ، وتبتدئ بالتلاؤم driverName و taxiId ، و numPass .

ب .

طبق بلغة جافا في الفئة Taxi طريقة taxiBusy التي تعدّل سيارة الأجرة لتكون مشغولة .

ج.

يريدون في الشركة تعريف الفئة محطة سيارات أجرة (TaxiStation) ، التي تستعمل الفئة سيارة أجرة (Taxi). تتبع للمحطة 80 سيارة أجرة على الأكثر.

صفات الفئة TaxiStation هي :

- اسم المحطة
- عدد سيارات الأجرة التابعة للمحطة .
- مصفوفة سيارات الأجرة التابعة للمحطة

سيارات الأجرة التابعة للمحطة ممثلة في مصفوفة سيارات الأجرة بصورة تسلسلية من بداية المصفوفة .

طبق بلغة جافا الفئة TaxiStation . تطبيق الفئة يشمل فقط البنود i-iii التي أمامك:

- تعريف صفات الفئة .
- الطريقة التي تُضيف سيارة أجرة إلى المحطة . تتلقى الطريقة اسم السائق ورقم الترخيص وعدد مقاعد الركاب .
- افترض أنّ عدد سيارات الأجرة التابعة للمحطة أقل من 80 .
- طريقة تتلقى عدد الركاب ، وتعيد رقم ترخيص سيارة الأجرة غير المشغولة
- الملائمة لنقل عدد الركاب المطلوب، وتعّدّل سيارة الأجرة لتكون مشغولة . إذا لم توجد سيارة أجرة غير مشغولة ملائمة، تعيد 1- .

**حل سؤال 3 :**

```
public class Taxi
{
    private int taxiId;
    private String driverName;
    private int numPass;
    private boolean available = true ;

    public Taxi(int tId, String dName)
    {
        this.taxiId = tId;
        this.driverName = dName;
        this.numPass = 4;
    }

    public Taxi(int tId, String dName, int nPass)
    {
        this.taxiId = tId;
        this.driverName = dName;
        this.numPass = nPass;
    }

    public int getnumPass()
    {
        return this.numPass;
    }

    public int gettaxiId()
    {
        return this.taxiId;
    }

    public boolean getavailable()
    {
        return this.available;
    }

    public void Busy()
    {
        this.available = false;
    }
}
```

```
public class TaxiStation
{
    private String StationName;
    private static int currentT=0;
    private Taxi T[] = new Taxi[80];

    public void AddTaxi(String dName, int tId, int nPass)
    {
        this.T[currentT++] = new Taxi(tId, dName, nPass);
    }

    public int checkAvailable(int N)
    {
        for (int i = 0; i < currentT; i++)
            if((T[i].getnumPass() >= N) && (T[i].getavailable() != false))
            {
                T[i].Busy();
                return (T[i].gettaxiId());
            }
        return -1;
    }
}
```

**سؤال 4 :**

الفئة Rational تُعرّف نمط المعطيات كسر (عدد نسبي) . يُمثّل الكسر بواسطة بسطه وبواسطة مقامه .

أمامك جزء من طرق الفئة Rational :

Rational(int n1, int n2)	طريقة بنائية تعيد كسراً بسطه n1 ومقامه n2 .
int getN1()	طريقة تعيد بسط الكسر
int getN2()	طريقة تعيد مقام الكسر

الفئة G تُعرّف نمط المعطيات مجموعة، الذي هو جمع كائنات من نمط Rational. عدد الكائنات في المجموعة غير ثابت . لا توجد أهمية لتسلسل ظهور الكائنات في المجموعة .

أمامك جزء من طرق الفئة G :

G()	طريقة تبني مجموعة فارغة
void gInsert(Rational num)	طريقة تُدخل الكسر num إلى المجموعة
Rational gRemove()	طريقة تعيد كسراً ما موجوداً في المجموعة وتحذفه من المجموعة
boolean gEmpty()	طريقة تعيد "صدق" إذا كانت المجموعة فارغة , أو "كذب" خلاف ذلك

طبق بلغة جافا في الفئة G, الطريقتين اللتين في البندين أ و ب .

يمكن استعمال الطرق المعطاة للفئتين Rational و G بدون تطبيقها .

أ .

G buildG(int n , int m)	طريقة تتلقى عدداً صحيحاً وموجباً n ، وعدداً صحيحاً وموجباً m . تعيد الطريقة مجموعة تشمل جميع الكسور التي بسطها هو عدد بين 1 و n ( بما في ذلك 1 و n ) ومقامها هو m .
-------------------------	--



ب.

Rational sum(int m)	<p>طريقة تتلقى عدداً صحيحاً <math>m</math> , وتعيد كسراً هو عبارة عن مجموع الكسور التي في المجموعة والتي مقامها هو <math>m</math> .</p> <p>افترض أنه يوجد في المجموعة على الأقل كسر واحد مقامه <math>m</math> .</p> <p>ملاحظة : بسط الكسر المعاد هو مجموع بسوط هذه الكسور , ومقامه هو <math>m</math> .</p>
---------------------	--

نسخة تجريبية - ليست للنشر

**حل سؤال 4:**

ملاحظة : داخل الفئة معرفّة قائمة بإسم Gr1 من نوع List<Rational> .  
الفرضية أننا نستطيع استعمال كل الطرق والدوال في الفئة Rational والفئة G بدون كتابتها .

```
List<Rational> Gr1 = new List<Rational>();
```

قسم أ :

```
public G BuildG(int n, int m)
{
    G gr2 = new G();
    while (! Gr1.GEmpty())
    {
        Rational r1 = Gr1.GRemove();
        if ((r1.GetN1() <= n) && (r1.GetN2() == m))
            gr2.GInsert(new Rational(r1.GetN1(), m ));
    }
    return gr2;
}
```

قسم ب :

```
public Rational Sum(int m)
{
    int sum = 0;
    while (!Gr1.GEmpty())
    {
        Rational r1 = Gr1.GRemove();
        if (r1.GetN2() == m)
            sum += r1.GetN1();
    }
    return (new Rational(sum, m));
}
```

**سؤال 5 :**

معطاة أربع الفئات : Test , MultiOne , SingleOne , Basis .  
تتبع بواسطة جدول متابعة العملية main في الفئة Test , واكتب المخرج .  
على الجدول أذ يشمل قيم جميع المتغيرات , وبالنسبة لكل كائن - قيم صفاته

```
public class Basis
{
    protected int num1;
    public Basis ()
    {
    }
    public Basis (int n)
    {
        this.num1 = n ;
    }
    public void print ()
    {
        System.out.println(this.num1);
    }
}

public class SingleOne extends Basis
{
    protected int num2;

    public SingleOne (int n1 , int n2)
    {
        super(n1);
        this.num2 = n2;
    }

    public void Print()
    {
        super.print();
        System.out.println(this.num2);
    }
}
```



```
}
```

```
public class MultiOne extends Basis
{
    private int count = 0;
    private Basis[] arr;

    public MultiOne ()
    {
        this.arr = new Basis[5] ;
    }

    public void Print ()
    {
        for (int i=0 ; i <= count ; i++ )
            arr[i].print ();
    }
    public void Add (Basis b)
    {
        arr[count] = b;
        count++;
    }
}
```

```
public class JavaApplication2
{
    public static void main(String[] args)
    {
        MultiOne container = new MultiOne();
        SingleOne s1 = new SingleOne(11, 35);
        container.Add(s1);

        s1 = new SingleOne(47, 22);
        container.Add(s1);

        s1 = new SingleOne(8, 17);
        container.Add(s1);

        MultiOne subContainer = new MultiOne();
        s1 = new SingleOne(53, 40);
        subContainer.Add(s1);

        s1 = new SingleOne(21, 13);
        subContainer.Add(s1);

        s1 = new SingleOne(39, 62);
        subContainer.Add(s1);

        container.Add(subContainer);
        container.Print();
    }
}
```

}

**حل سؤال 5:**

نتيجة الطباعة : 11 35 47 55 8 17 53 40 21 13 39 62

الميزة :	خطوة 1	2	خطوة 3	4	خطوة 5	6	خطوة 7	8	9	10	11	12
container:count	0		1		2		3					
Container:arr[0]	null		address of SingleOne object, with num1=11 num2=35		address of SingleOne object, with num1=11 num2=35		address of SingleOne object, with num1=11 num2=35					
Container:arr[1]	null		null		address of SingleOne object, with num1=47 num2=22		address of SingleOne object, with num1=47 num2=22					
Container:arr[2]	null		null		null		address of SingleOne object, with num1=8 num2=17					
Container:arr[3]	null		null		null		null					
Container:arr[4]	null		null		null		null					
Container:num1	0											
s1 : num1		11		47		8						
s2 : num2		35		22		17						
SubContainer: arr[0]												
SubContainer: arr[1]												
SubContainer: arr[2]												
SubContainer: arr[3]												
SubContainer: arr[4]												
SubContainer: num1												

الميزة	1	2	3	4	5	6	7	8	9	خطوة 10	11	خطوة 12	13	خطوة 14	خطوة 15
container:count															4
Container:arr[0]															
Container:arr[1]															
Container:arr[2]															
Container:arr[3]															address of MultiOne object, SubContai ner
Container:arr[4]															
Container:num1	0														
s1 : num1		1		4		8			5		21		39		
s2 : num2		3		2		1			4		13		62		
SubContainer: count		5		2		7			0		1		2		3
SubContainer: arr[0]								null							address of SingleOne object, with num1=53 num2=40
SubContainer: arr[1]								null							address of SingleOne object, with num1=21 num2=13
SubContainer: arr[2]								null							address of SingleOne object, with num1=39 num2=62
SubContainer: arr[3]								null							null
SubContainer: arr[4]								null							null
SubContainer: num1								0							

**سؤال 6 :**

أمامك مشروع فيه واجهات التطبيق IFirst , ISecond, IThird , والفئات AAA, BBB, CCC والفئة Test .

```
public interface IFirst
{
    public boolean OpA (Object stam);
    public void OpB (int num);
}
//-----
public interface ISecond extends IFirst
{
    public int OpC ();
}
//-----
public interface IThird
{
    public int OpD ();
}
//-----
public class AAA implements IFirst
{
}
//-----
public class BBB implements ISecond
{
}
//-----
public class CCC implements ISecond, IThird
{
}
//-----
public class Test
{
    public static void main(String[] args)
    {
    }
}
```

أ) أكتب أيّة عمليات يجب تطبيقها في كلّ واحدة من الفئات AAA, BBB, CCC , وإشرح لماذا .

ب) لكل واحدة من مجموعات الأوامر i-iv التي أمامك , حدّد إذا كانت قانونية أم لا ؟ علّل

- i. ISecond s1 = new ISecond ();
- ii. BBB b1 = new BBB ();
- iii. AAA a1 = new AAA ();  
IFirst f1 = a1 ;
- iv. CCC c1 = new CCC ();  
IFirst f2 = c1 ;

ج. أمامك المتطلّبان الآتيان:

- تشغيل العملية OpB على الكائن من النمط BBB .
- تحويل كائن من النمط CCC ليصبح كائناً من النمط AAA.

لكل واحد من المتطلّبين , حدّد إذا كان من الممكن تنفيذه بواسطة كتابة أمر في العملية main.

إذا كان من الممكن تنفيذه – اكتب الأمر الملائم .

إذا لم يكن من الممكن تنفيذه – علّل لماذا .



**حل سؤال 6:**

دوال وطرق للتطبيق داخل- AAA

```
public bool opA(Object stam)
public void opB (int num)
```

دوال وطرق للتطبيق داخل - BBB

```
public bool opA(Object stam)
public void opB (int num)
public int opC()
```

دوال وطرق للتطبيق داخل - CCC

```
public bool opA(Object stam)
public void opB (int num)
public int opC()
public int opD()
```

ب.

- (i) غير قانوني. نحاول بناء كائن من نوع واجهة تطبيق وهذا خاطئ.
- (ii) قانوني. استدعاء العملية البنائية الافتراضية للفئة BBB وبناء من نوع BBB.
- (iii) قانوني. بناء كائن من نوع AAA ويؤشر عليه مؤشر من نوع AAA. نحول بعدها المؤشر الى نوع Ifirst (قانوني لأنه يطبق الواجهة Ifirst)
- (iv) قانوني. لأن CCC تطبق الواجهات ISecond و- IThird و- ISecond يرث Ifirst يوجد هنا upcasting وهذا قانوني.

ج.

```
public static void main (String[] arg)
{
    BBB b = new BBB();
    b.opB(3);

    Object c = new CCC();
    AAA a = (AAA) c;
}
```

يوجد تحويل, لكن عند التشغيل نحصل على أخطاء وقت التشغيل .RunTime Error

**سؤال 7:**

الفئة Point تُعرّف نمط المعطيات نقطة في المستوى .توصّف النقطة في هيئة محاور بواسطة إحداثيّها (x , y) . x و y هما عدنان صحيحان .

للفئة Point الصفتان التاليتان :

int x , int y

أمامك جزء من واجهة تطبيق الفئة Point :

Point (int x, int y)	عملية بنائية, تضع القيم في الميزات ( x, y )
int getX()	عملية تعيد قيمة x للنقطة
int getY()	عملية تعيد قيمة y للنقطة

الفئة Line تُعرّف نمط معطيات قطعة في المستوى . تُعرّف قطعة في المستوى بواسطة النقطتين الموجودتين في طرفيها .

للفئة Line الصفتان التاليتان:

Point point1

Point point2

أمامك جزء من واجهة تطبيق الفئة Line :

Line (Point p1, Point p2)	عملية بنائية تُعيد قطعة في المستوى موجودة بين النقطتين p1 و p2 .
Point GetPoint1()	عملية تُعيد النقطة point1 للقطعة .
Point GetPoint2()	عملية تُعيد النقطة point2 للقطعة .

الفئة Drawing رسم، هي مجموعة نقاط في المستوى ومجموعة قطع في المستوى .  
للفئة Drawing الصفات التالية :

- مصفوفة نقاط
- مصفوفة قطع .
- عدد النقاط الموجودة في الرسم .
- عدد القطع الموجودة في الرسم .

طبق بلغة جافا البنود "أ" – "و" في الفئة Drawing . استعمل واجهتي تطبيق الفئتين Line و Point بدون تطبيقهما . يمكن إضافة عمليات لواجهتي التطبيق هاتين . إذا أضفت عمليات عليك تطبيقها .

أ . عنوان الفئة وصفاتها .

ب . عملية بنائية تتلقى عدد النقاط التي يمكنها أن تكون في الرسم - np, وعدد القطع التي يمكنها أن تكون في الرسم - n1.

تنفيذ العملية :

- ابتداء مصفوفة النقاط لتكون بـ np
- ابتداء مصفوفة القطع لتكون بـ n1
- ابتداء عدد النقاط الموجودة في الرسم إلى 0 .
- ابتداء عدد القطع الموجودة في الرسم إلى 0.

ج . عملية تَعِيد النقطة الموجودة في المكان n في مصفوفة نقاط الرسم . افترض أنه توجد نقطة في المكان n في المصفوفة.

د . عملية تضيف نقطة إلى الرسم افترض أنه يوجد مكان للنقطة في مصفوفة النقاط .

هـ . عملية تتلقى نقطة في الرسم، وتُعِيد عدد القطع التي توجد النقطة في أحد أطرافها .

و . عملية تتلقى نقطة في الرسم، وتعيد "صدق" - إذا لم تكن أية قطعة، هذه النقطة هي أحد أطرافها، و "كذب" - خلاف ذلك .

**حل سؤال 7:**

```
class Drawing
{
    private Line[] lines;
    private Point[] points;
    private int countpoints = 0;
    private int countLines = 0;

    public Drawing(int np, int nl)
    {
        lines = new Line[nl];
        points = new Point[np];
    }

    public Point getPoint(int num)
    { return points[num]; }

    public void addPoint(Point p)
    { points[countpoints++] = p; }

    public int findPoints(Point p)
    {
        int sum = 0;
        for (int i = 0; i < countLines; i++)
        {
            Point q = lines[i].getPoint1();
            if((p.getX()==q.getX()) && (p.getY()==q.getY()))
                sum++;
            q = lines[i].getPoint2();
            if ((p.getX()==q.getX()) && (p.getY()==q.getY()))
                sum++;
        }
        return sum;
    }

    public boolean noLineWithPoint(Point p)
    {
        return findPoints(p) == 0;
    }
}
```

**سؤال 8 :**

قرّر اتحاد كرة السلة حوسبة تسجيل لاعبي كرة السلة في الدولة . لهذا الغرض عرّف الإتحاد ثلاث فئات : لاعب كرة سلة (Player) ، وفريق (Team) ، واتحاد كرة السلة (Union) .

الفئة Player الصفات التالية :

- اسم لاعب كرة السلة - String playerName
- طول لاعب كرة السلة بالأمتار - double height
- عدد النقاط التي أحرزها في هذا الموسم - int points

في الفئة Player عُرّفت العمليات المُعيدة :  
getPlayerName() , getHeight() , getPoints()  
والعمليات المحددة :

setPlayerName(String name) , setHeight(double height) ,  
setPoints(int points)

الفئة Team الصفات التالية :

- اسم الفريق - String teamName
- عدد اللاعبين في الفريق - int num
- مصفوفة بـ 20 للاعبي الفريق - Player[] playerArray

في الفئة Team عُرّفت العمليات المُعيدة :  
getTeamName() , getNum() , getPlayerArray()  
والعملية المحددة :  
setTeamName(String name)

طبق بلغة جافا الفئة Union، التي تستعمل الفئتين Player و Team. في اتحاد كرة السلة يوجد 30 فريقاً .

الفئة Union الصفة : مصفوفة فرق كرة السلة .  
يشمل تطبيق الفئة البندين الفرعيين i-ii اللذين أمامك :  
i. عنوان الفئة  
ii. تعريف صفة الفئة

ب. طبق بلغة جافا العملية التي أمامك في الفئة Union.

Void printTeamsDetails()	تطبع العملية لكل فريق في اتحاد كرة السلة، اسم الفريق وعدد اللاعبين فيه.
--------------------------	---

ج) يعني اتحاد كرة السلة بالحصول على معطيات عن عدد النقاط التي أحرزها كل واحد من اللاعبين في الموسم الماضي. لمعرفة كم لاعباً ضعيفاً , وكم لاعباً متوسطاً , وكم لاعباً ممتازاً في إحراز النقاط يوجد في الاتحاد.

اللاعب الضعيف في إحراز النقاط هو لاعب أحرز في الموسم أقل من 100 نقطة .  
اللاعب المتوسط في إحراز النقاط هو لاعب أحرز في الموسم بين 100 و 200 نقطة  
(بما في ذلك 100 و 200) .

اللاعب الممتاز في إحراز النقاط هو لاعب أحرز في الموسم أكثر من 200 نقطة.

اكتب العمليات التي يجب إضافتها إلى كل واحدة من الفئات الثلاث Team و Player و Union . للحصول على المعلومات المطلوبة . في كل واحدة من المئات عليك إضافة عملية واحدة على الأقل . بالنسبة لكل عملية أضفتها، اكتب توقيعها بلغة جافا، وتوثيقها، وفي أية فئة يجب تطبيقها .

**حل سؤال 8 :**

```
public class player
{
    private String playerName;
    private double height;
    private int points;

    public player(String pN, double h, int pont)
    {
        this.playerName = pN;
        this.height = h;
        this.points = pont;
    }
    public void SetPlayerName(String pN)
    {
        this.playerName = pN;
    }
    public String GetPlayerName()
    {
        return this.playerName;
    }

    public void SetHeight(double height)
    {
        this.height = height;
    }
    public double GetHeight()
    {
        return this.height;
    }
    public void SetHeight(int pont)
    {
        this.points = pont;
    }
    public int GetPoints()
    {
        return this.points;
    }
    @Override public String toString()
    {
        String st = "";
        st += "PlayerName:" + this.playerName + "\n";
        st += "Height:" + this.height + "\n";
        st += "Points:" + this.points + "\n";
        return st;
    }
}
```

```
public class Team
{
    private String teamName;
    private int num;
    private player[] playersArray = new player[20];

    public Team(String TN, int n1)
    {
        this.teamName = TN;
        this.num = n1;
    }
    public void SetTeamName(String TN)
    {
        this.teamName = TN;
    }
    public String GetTeamName()
    {
        return this.teamName;
    }
    public void SetNum(int n1)
    {
        this.num = n1;
    }
    public int GetNum()
    {
        return this.num;
    }
    public void SetPlayersArray(player [] arr)
    {
        this.playersArray = arr;
    }
    public player[] GetPlayersArray()
    {
        return this.playersArray;
    }
    @Override public String toString()
    {
        return "\nTeam:" + this.teamName+" No.Players:" + this.num;
    }
}
```

```
public class Union
{
    private Team[] TArr = new Team[30];
    public Union()
    {
    }
    public void PrintTeamsDetails()
    {
        for (int i = 0; i < TArr.length; i++)
            if (TArr[i] != null)
                TArr[i].toString();
    }
}
```



ج) هنالك عدّة طرق للحل , لكن طُلب في السؤال أن نضيف دالة بكل واحدة من الفئات .  
الاقتراح :  
أن تكتب دالة داخل الفئة `player` , تعيد قيمة حسب عدد النقاط التي أحرزها اللاعب.

```
public int GetStatus()
{
    if (this.points < 100) return 0;
    else if ((this.points >=100) && (this.points <= 200)) return 1;
    else return 2;
}
```

وأيضاً دالة تكتب داخل الفئة `Team` , نُعرّف مصفوفة بحجم 3 مصفوفة عدادات .  
نمشط المصفوفة `playersArray` مع مناداه للدالة `GetStatus` وحسب القيمة المعادة  
(0,1,2) نعدل المكان في المصفوفة العدادات .

```
public int[] GetTeamStatus()
{
    int[] A = new int[3];
    for (int i = 0; i < this.num; i++)
    {
        int j = this.playersArray[i].GetStatus();
        A[j]++;
    }
    return A;
}
```

وأيضاً دالة تكتب داخل الفئة `Union` , نُعرّف 3 متغيرات (`c1`, `c2`, `c3`) , نمشط كل المجموعات وحسب مصفوفة العدادات الذي نحصل عليه من الدالة `GetTeamStatus` , نعدل الثلاثة متغيرات.

```
public void PrintTeamsStatus()
{
    int[] B;
    int c1=0, c2=0, c3=0;
    for (int i = 0; i < TArr.length; i++)
    {
        if (TArr[i] != null)
        {
            B = TArr[i].GetTeamStatus();
            c1 += B[0];
            c2 += B[1];
            c3 += B[2];
        }
    }
    System.out.println("\nNo. of Weeks: " + c1);
    System.out.println("\nNo. of Acceptable: " + c2);
    System.out.println("\nNo. of Goods: " + c3);
}
```

**سؤال 9 :**

في موقع "الإستطلاع الأسبوعي" يُنشر كل أسبوع استطلاع جديد فيه سؤال واحد . يحفظ الموقع الإستطلاعات ال- 50 الأخيرة , بما في ذلك الاستطلاع الجديد . معطيات كل استطلاع هي : تاريخ نشره , السؤال الذي يُطرح فيه , أربع إجابات ممكنة عن السؤال , وبالنسبة لكل إجابة - عدد المشتركين الذين اختاروها . بإمكان المشترك الإجابة فقط عن السؤال في الاستطلاع الأخير الذي نُشور, ويجوز له اختيار إجابة واحدة فقط .

(أ) عرّف الفئات اللازمة لتطبيق الموقع "الإستطلاع الأسبوعي" .  
بالنسبة لكل واحدة من الفئات التي تعرّفها :

- i. اكتب ماذا تمثّل .
- ii. اكتب عنوانها بلغة Java .
- iii. عرّف صفاتها بلغة Java , و اكتب توثيقاً لكل صفة .

(ب) . إدارة الموقع "الإستطلاع الأسبوعي" تشمل :

- إضافة استطلاع جديد
- طباعة سؤال الاستطلاع الجديد , وطباعة الإجابات الأربع الممكنة عنه .
- استقبال الإجابة التي اختارها المشترك في الاستطلاع الجديد , وتعديل عدد المشتركين الذين اختاروا هذه الإجابة .
- استقبال تاريخ نشر الاستطلاع , وبالنسبة لسؤال الاستطلاع الذي نشر في هذا التاريخ , طباعة المعطيين التاليين:

- ✓ عدد المشتركين الذين اختاروا كل واحدة من الإجابات .
- ✓ ماذا كانت الإجابة التي اختارها أكبر عدد من المشتركين .

عرّف العمليات اللازمة لإدارة الموقع "الاستطلاع الأسبوعي".  
افتراض أن العمليتين `get` و `set` معرفتان بالنسبة لكل صفة في كل واحدة من الفئات التي عرفتها في البند أ .

بالنسبة لكل واحدة من العمليات , اكتب :

- i. في أي فئة يجب تعريفها .
- ii. عنوانها بلغة Java.
- iii. توثيقاً لها .

ج.

طبق بلغة Java عملية تعيد عدد الاستطلاعات التي اشترك فيها أكثر من 1000 مشترك. اذكر في أي فئة من الفئات التي عرفتها في البند "أ" يجب تطبيق هذه العملية . افتراض أن العمليتين `get` و `set` معرفتان بالنسبة لكل صفة في كل واحدة من الفئات التي عرفتها في البند "أ".

**حل سؤال 9 :**

أ. فئة تمثل إستطلاع واحد :

```
public class OneOpinionPoll
{
    // تاريخ نشر الاستطلاع بالشكل dd/mm/yyyy
    private string date ;
    private string question ; // سؤال الاستطلاع
    // الاجوبة لسؤال الاستطلاع
    private String[] answer = new String [4];

    // مصفوفة كل خليه تمثل كمية
    private int[] numberOfAnswers = new int [4] ;

    // مشتركون اختاروا جواب معين
    .....
}
```

2. فئة تمثل 50 الاستطلاعات الاخيرة :

```
public class FiftyOpinionPolls
{
    private OneOpinionPoll[] opinionPolls=new OneOpinionPoll [50];
    // مصفوفة ل 50 الاستطلاعات الاخيرة
    private int indexOfNewestOpinionPoll ;
    // الرقم التسلسلي للأستطلاع الجديد من بين ال 50 استطلاع
    ...
}
```

ملاحظة :

نفترض أن 50 الاستطلاعات الاخيرة خُزِنَتْ وحُفِظَتْ .  
لمعالجة 50 الاستطلاعات يجب معرفة اماكن الاستطلاع الجديد والقديم .  
اذا كان الرقم التسلسلي indexOfNewestOpinionPoll مساوٍ ل 49 اذا الاستطلاع القديم  
الذي خزن داخل المصفوفة موجود في المكان 0 .  
خلاف ذلك حسب  $indexOfNewestOpinionPoll + 1$  .

1. في الفئة OneOpinionPoll :

الطريقة – الدالة تتلقى رقم الجواب للمشارك في الاستطلاع (1-4) كبارامتر وتقوم بزيادة عدد المشتركين لهذا الجواب بواحد .

```
Public void incNumberOfAnswers( int i )
```

(ب) في الفئة FiftyOpinionPolls :

الدالة تستقبل رقم الجواب المشترك بالاستطلاع الاخير (الجديد) كبارامتر وتقوم بزيادة عدد المشتركين لهذا الجواب بواحد .

```
public void incNumberOfAnswers(int i)
```

الدالة تتلقى تاريخ نشر الاستطلاع date كبارامتر dd/mm/yyyy , وتتلقى سؤال الاستطلاع q , ومصنوفة بكم 4 هي الاجوبة للاستطلاع m .  
الدالة تقوم بتبديل الاستطلاع القديم (ذو الاقدمية الاكبر) باستطلاع جديد تكون معطياته 0 اختيارات للأجوبة الاربعة .  
بالاضافة الدالة تعدّل الرقم التسلسلي للاستطلاع الجديد .

```
public void UpdateOpinionPoII(string date,string q, String[] m)
```

الدالة تطبع السؤال والاجوبة للاستطلاع الجديد :

```
public void printNewestOpinionPoll () ;
```

الدالة تتلقى تاريخ نشر الاستطلاع date كبارامتر dd/mm/yyyy , وتطبع سؤال الاستطلاع الذي نشر بهذا التاريخ , وعدد المشتركين الذين اختاروا بكل واحدة من الاجوبة الاربعة, ورقم الجواب الذي أختير بواسطة الأكثرية من المشتركين .

```
public void printOpinionPoll (string date)
```

(ج) في الفئة FiftyOpinionPolls :

```
public int numOfOpinionPolls()
```

```
{
    int counter=0 ;
    for(int i=0; i<50; i++ )
    {
        int num = opinionPolls[i].sum();
        if ( num>1000 ) counter++ ;
    }
    return counter ;
}
```

الدالة numOfOpinionPolls تستعمل الدالة Sum من الفئة OneOpinionPoll .  
الدالة تعيد عدد المشتركين الذين اختاروا الاجوبة للاستطلاع الحالي

```
public int sum ()
```

```
{
    int help=0 ;
    for (int i = 0 ; i < 4; i++)
        help += numberOfAnswers [ i ] ;
    return help
}
```

**سؤال 10 :**

في المتحف "القديم والجديد" طُوّر مشروع لتسجيل المعروضات في المتحف.  
أمامك تعريف جزئي للفئات بلغة Java:

معروض- Exhibit , رسمة- Painting , تمثال- Statue , صورة- Photograph.

```
public class Exhibit
{
    protected string creator;    // اسم المبدع
    protected string name;       // اسم المعروض
    protected int year;          // اسم العمل الفني
    protected double height;     // طول المعروض
    protected double width;      // عرض المعروض

    public Exhibit()
    {
        //...
    }

    public Exhibit(string creator, string name, int year,
                   double h, double w)
    {
        //...
    }

    /** العملية تُعيد مساحة العرض اللازمة لعرض المعروض في المعرض */
    public virtual double ExhibitionSpace()
    {
        return this.height * this.width;
    }
}

public class Painting extends Exhibit
{
    private string style;        // اسلوب الرسمة
}
```

```

public class Statue extends Exhibit
{
    private string[] materials; // المواد التي تبني منها التمثال
    private string technique; // تقنية النحت
    private double depth; // عمق التمثال
    public override double ExhibitionSpace()
    {
        return this.height * this.width * this.depth;
    }
}

public class Photograph extends Exhibit
{
    private string kind; // نوع الصورة
}

```

أ. طبق بلغة Java عملية بائية في الفئة Painting ، تستعمل العملية البائية التي تتلقى بارامترات في الفئة Exhibit.

ب. اذكر أية آلية برمجة موجهة كائنات تنعكس في تعريف العملية exhibitionSpace() التي في الفئة Exhibit والعملية exhibitionSpace() التي في الفئة Statue.

ج. هل يمكن تطبيق العملية exhibitionSpace() في الفئة Statue بمساعدة العملية exhibitionSpace() التي في الفئة Exhibit ؟ إذا كاذ يمكن – طبقها؟ وإذا لم يكن ممكناً - علل لماذا ؟

د. الفئة غرفة - Room ، تمثل غرفة في المتحف . ني الغرفة يمكن أن يكون حتى 25 معروضاً . في الغرفة يمكن أن تكون رسومات ، تماثيل، صور

طبق بلغة Java الفئة غرفة - Room . يجب أن يشمل تطبيق الفئة البنود i-iii التي أمامك :

- i. عنوان الفئة , وتعريف صفاتها .
- ii. عملية تعيد عدد الرسومات الموجودة في الغرفة .
- iii. عملية تتلقى معروضاً وتضيفه الى الغرفة . افترض أنه يوجد لهذا المعروض مكان في الغرفة .



(هـ)  
الفئة معرض Exhibition، تمثل معرضاً فنياً يُقام في المتحف في 10 غرف  
مرقّمة من 1 حتى 10.

طبّق بلغة Java الفئة Exhibition، يجب أن يشمل تطبيق الفئة البنود i-iii  
التي أمامك :

- i. عنوان الفئة ، وتعريف صفاتها .
- ii. عملية تتلقّى رقم غرفة بين 1 و 10 ، وتعيد عدد الرسومات الموجودة في هذه الغرفة.
- iii. عملية تتلقّى رقم غرفة بين 1 و 10 ، ومعرضاً ، وتضيف المعرض إلى الغرفة الملائمة . افترض أنّه يوجد لهذا المعرض مكان في الغرفة .



**حل سؤال 10 :**

```
public class Painting(String creator, String name, int year,
double height, double width, String style)
{
    super(creator, name, year, height, width);
    this.style = style;
}
```

ب. دهرس (overriding) - تعريف من جديد للدالة او العملية داخل الفئة الوارثة .  
ج.

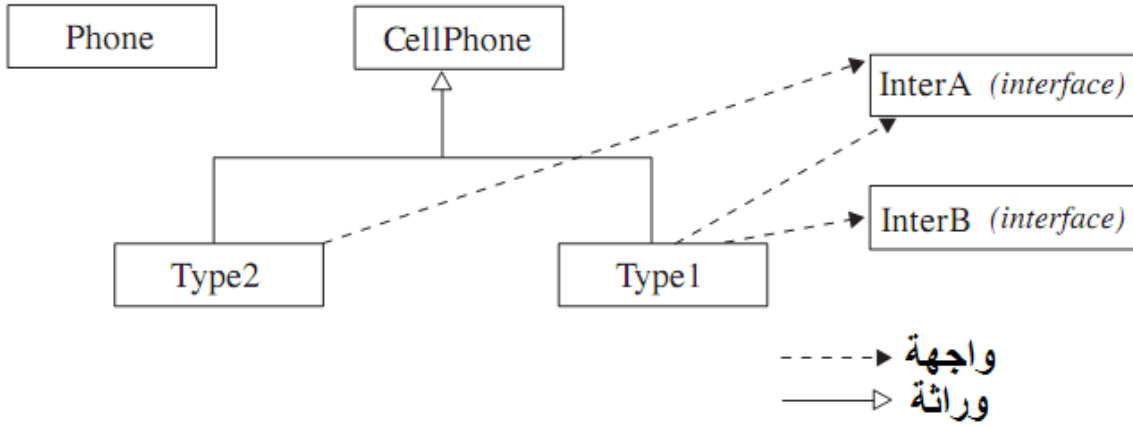
```
public double exhibitionSpace ()
{
    return super.exhibitionSpace () * this.depth;
}
```

```
//-----
public class Room //صفوفة المعارضات بـ 25
{
    private Exhibit [] items = new Exhibit[25];
    private int num; // عدد المعارضات بالغرفة
    public int numOfPaints()
    {
        int counter=0 ;
        for (int i=0; i < num; i++ )
            if ( items[i] instanceof Painting ) counter++ ;
        return counter ;
    }
    public void addItem ( Exhibit ex )
    {
        items[num]=ex ;
        num++ ;
    }
    .....
} // نهاية الفئة Room
```

```
public class Exhibition
{
    private Room[] rooms=new Room [10] ; // غرف العرض
    public int numOfPaints (int i)
    {
        return rooms[ i - 1 ].numOfPaint() ;
    }
    public void addItem ( int i, Exhibit ex)
    {
        rooms[ i - 1].addItem (ex) ;
    }
    .....
} // نهاية الفئة Exhibition
```

**سؤال 11 :**

تطور شركة هواتف مشروعاً معيناً  
التخطيط الذي أمامك هو وصف جزئي للفئات التي في المشروع



يوجد للشركة هواتف بيتية وهواتف خلوية  
الفئة Phone تمثل هاتفاً بيتياً ، والفئة CellPhone تمثل هاتفاً خلوياً .  
توجد هواتف خلوية من نوعين : Type1 , Type2 .  
أمامك معلومات عن العمليات التي يمكن إجراؤها في كل واحد من الأنواع

في الهاتف البيتي:

- اتصال
- تلقي مكالمة

في الهاتف الخلوي من النوع Type1:

- اتصال
- تلقي مكالمة
- إدارة ذاكرة أرقام الهواتف : إضافة رقم، محو رقم .
- التقاط صورة
- عرض صورة
- عرض الساعة
- ضبط الساعة

في الهاتف الخليوي من النوع Type2:

- اتّصال
- تلقّي مكالمة
- إدارة ذاكرة أرقام الهواتف : إضافة رقم، محو رقم .
- عرض الساعة
- ضبط الساعة

في واجهة التطبيق IntraA معرفة العمليات اللتان تعالجان الساعة : عرض الساعة، ضبط الساعة .

في واجهة التطبيق IntraB معرفة العمليات اللتان تعالجان الصور: التقاط صورة، عرض صورة.

(أ)

لتمكين إعادة استعمال الكود (code reuse), يجب إضافة فئة إلى المشروع فيها معالجة لكل الصفات والعمليات المشتركة بين الهواتف البيئية والهواتف الخلوية بالنسبة لهذه الفئة :

- i. اكتب عنوان الفئة بلغة Java .
- ii. اكتب بالكلمات العمليات التي يجب أن تُشمل فيها , وأيّة صفات يجب تعريفها في الفئة , بحيث يكون بالإمكان تطبيق هذه العمليات .

(ب)

انسخ التخطيط المعطى ( في الصفحة السابقة ) إلى دفترك , وأضف إليه الفئة التي عرّفتها في البند أ . ارسم في التخطيط الذي في دفترك الروابط الجديدة التي تكونت و اشرحها بالكلمات .

(ج) بافتراض أنهم أضافوا إلى المشروع الفئة التي عرّفها في البند "أ" بالنسبة لكل واحدة من الفئات التي في التخطيط ( لا يشمل واجهات التطبيق , ولا يشمل الفئة التي عرّفها في البند أ ) .

- i. اكتب عنوان الفئة بلغة Java.
- ii. اكتب بالكلمات العمليات التي يجب أن تُشمل فيها , وأيّة صفات يجب تعريفها في الفئة ء بحيث يكون بالإمكان تطبيق هذه العمليات .

(د) في حاسوب مركز خدمات الشركة , ممثلة الهواتف من كل الأنواع الموجودة في التصليح . يجب إضافة الفئة اللازمة لهذا التمثيل إلى المشروع. اكتب بلغة Java عنوان الفئة , وصفاتها , واكتب توثيقاً للصفات .

**حل سؤال 11 :**

أ. الفئة تمثل هاتف عادي (هاتف أساسي)

(i)

```
public class BasePhone
```

(ii)

الميزات :

```

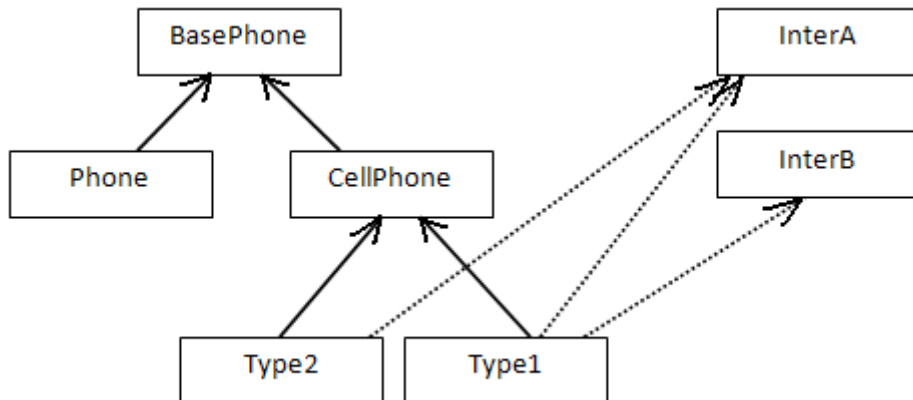
string numberOfPhone ; // رقم الهاتف
int status ;           // مكالمات واردة - 1 -
                      // مكالمات خارجية - 2 -
                      // يتصل - 3 -
                      // اتصال - 4 -
                      // متوفر

```

الدوال :

- اتصال
- تلقي مكالمات

ب .



الفئات Phone و CellPhone ترثان الفئة BasePhone .

ج .

(i)

```

public class Phone extends BasePhone
public class CellPhone extends BasePhone
public class Type1 extends CellPhone implements InterA, InterB
public class Type2 extends CellPhone implements InterA

```

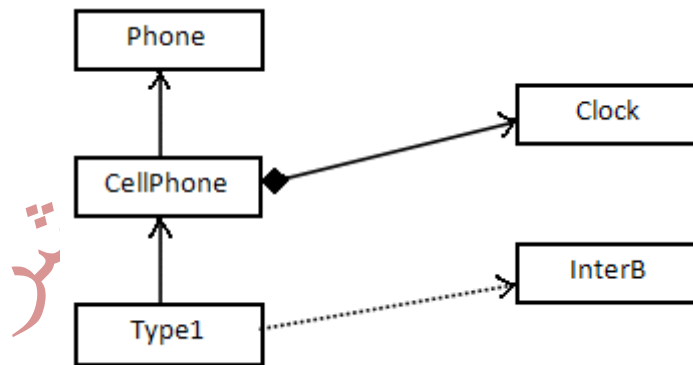
( ii ) الفئة Phone :

لا حاجة لإضافة ميزات أو دوال الى الفئة Phone كل الأشياء التي نحتاجها متوفرة داخل الفئة BasePhone .

الفئة	الميزة	دوال - طرق
CellPhone	مصفوفة لأرقام الهواتف	إضافة رقم حذف رقم
Type1		عرض الساعة تعديل الساعة تصوير صورة عرض الصورة
Type2		عرض الساعة تعديل الساعة

ملاحظة :

في فرع (أ) طلب فئة جديدة , اعتقد أن الواجهه InterA يجب أن تكون فئة .  
السبب : اذا وجدت عمليات أو طرق لتعديل الساعة , اذا هنالك حاجة لحفظ معطيات الساعة بالميزات.  
وحسب ما ذكر في السؤال يجب ترتيب الفئات حسب UML التالي :



الفئة CellPhone تحوي reference لكائن من نوع Clock , وترث من الفئة Phone.  
الفئة Type1 تطبق الواجهه InterB .

دوال - طرق	الميزة	الفئة
اتصال استقبال مكالمة	رقم الهاتف // string numberOfPhone ;  مكالمة واردة - 1 - // مكالمة خارجية - 2 - // يتصل - 3 - // اتصال - 4 - // int status;	Phone
إضافة رقم حذف رقم	مصفوفة لأرقام التلفونات ساعة	CellPhone
عرض الساعة تعديل الساعة	ساعات دقائق ثوان	Clock
تصوير صورة عرض الصورة		Type1

(د)

```

public class Service
{
    private ArrayList v ;           مجمع دينامي لجهاز الهاتف//
    .....
}

```

**سؤال 12 :**

أمامك برنامج وفيه الفئات التالية :

MainApp - و Derived3 , Derived2 , Derived1, Base

```
public class Base
{
    protected static int[] arr;
    protected Base successor;

    public void SetSuccessor(Base b)
    {
        this.successor = b;
    }

    public static void SetArr(int[] arr)
    {
        Base.arr = arr;
    }

    public void Opp()
    { System.out.println("Good Luck !"); }
}

public class Derived1 extends Base
{
    public void opp()
    {
        int sum = 0;
        for (int i = 0; i < Base.arr.Length; i++)
        {
            if (Base.arr[i] > 0)
            {
                sum += Base.arr[i];
            }
        }
        System.out.println ("Sum :" + sum);
        if (this.successor != null)
            this.successor.opp();
    }
}
```



```
public class Derived2 extends Base
{
    public void Opp()
    {
        int counter = 0;
        for (int i = 0; i < Base.arr.Length; i++)
        {
            if (Base.arr[i] <= 0)
            {
                counter++;
            }
        }
        System.out.println ("Counter : " + counter);
        if (this.successor != null)
            this.successor.Opp();
    }
}
```

```
public class Derived3 extends Base
{
    public void opp()
    {
        int counter = 0;
        for (int i = 0; i < Base.arr.Length; i++)
        {
            if (Base.arr[i] % 2 == 0)
            {
                counter++;
            }
            if (Base.arr[i] == 0)
                System.out.println ("*");
            else
                System.out.println (Base.arr[i]);
        }
        System.out.println ("Counter : " + counter);
        if (this.successor != null)
            this.successor.Opp();
    }
}
```

```

public class MainApp
{
    public static void main(String[] args)
    {
        Derived1 d1 = new Derived1();
        Derived2 d2 = new Derived2();
        Derived3 d3 = new Derived3();

        d1.SetSuccessor(d2);
        d2.SetSuccessor(d3);

        int[] arr = { 12, 13, -5, 0, -56, 34, 22, -30 };
        Base.SetArr(arr);
        d1.opp();
    }
}

```

اكتب متابعة للعملية main في الفئة MainApp، واكتب المخرج .  
يجب أن تكتب في المتابعة قيم المتغيرات، وبالنسبة لكل كائن قيم صفاته .

(ب)  
نستبدل العملية main التي في الفئة MainApp بالعملية main التي أمامك :

```

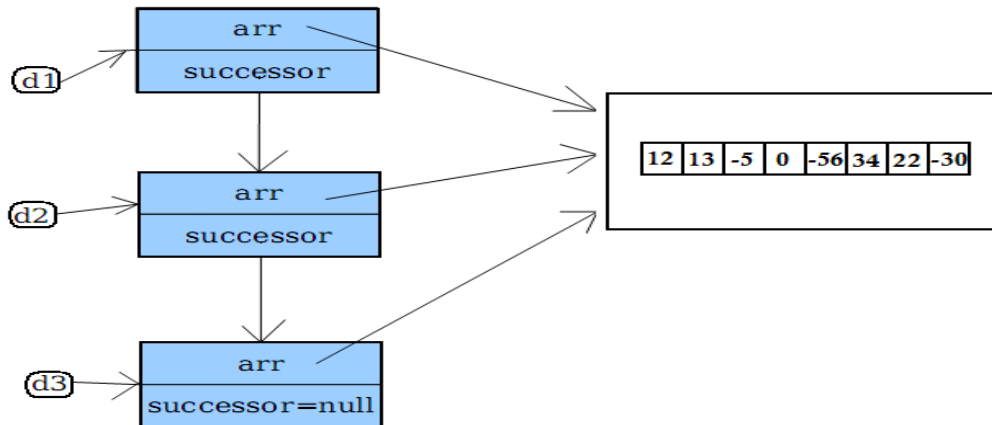
public static void Main(string[] args)
{
    Derived1 d4 = new Derived1();
    d4.SetSuccessor(d4);
    int[] arr = { 12, 13, -5, 0, -56, 34, 22, -30 };
    Base.SetArr(arr);
    d4.Opp();
}

```

أمامك الإدعاء : تنفيذ العملية main لا ينتهي .  
حدّد إذا كان هذا الإدعاء صحيحاً أم غير صحيح , علل ؟

**حل سؤال 12 :**

(أ)



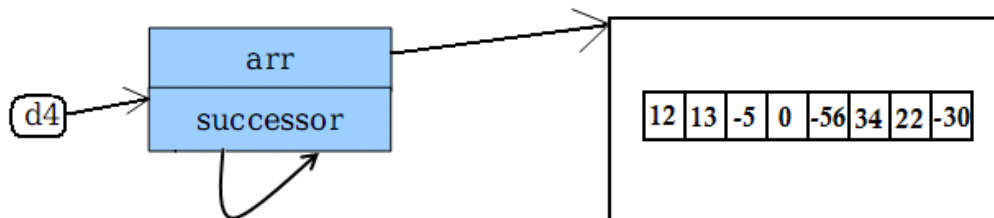
وصف للكائنات التي نتجت بعد تنفيذ البرنامج .

المخرج:

Sum: 81  
 Counter: 4  
 12  
 13  
 -5  
 \*  
 -56  
 34  
 22  
 -30  
 Counter: 6

(ب)

البرنامج لا ينتهي لأن المؤشر successor لا يمكن ان تكون قيمته null .



**سؤال 13 :**

أمامك جزء من مشروع يتناول وسائل مواصلات ويشمل الفئات التالية:

الفئة Vehicle	تمثل وسيلة مواصلات
الفئة Train	تمثل قطاراً الذي هو وسيلة مواصلات
الفئة Boat	تمثل قارباً الذي هو وسيلة مواصلات
الفئة Airplane	تمثل طائرة التي هي وسيلة مواصلات
الفئة TransportationCompany	تمثل شركة لديها وسائل مواصلات من أنواع مختلفة

```

public class Vehicle
{
    private String type;    // الموقع ( أرض / مياه / جو )
    private String way;    // نوع الطريق ( شارع / سكة حديد / نهر )
    private int maxSpeed;  // السرعة القصوى
    public Vehicle(String type, String way, int maxSpeed)
    {
        this.type = type;
        this.way = way;
        this.maxSpeed = maxSpeed;
    }
}

//-----
public class Train extends Vehicle
{
    private int numOfCarriages;    // عدد عربات القطار
    public Train(int maxSpeed, int numOfCarriages)
    {
        super("land", "tracks", maxSpeed);
        this.numOfCarriages = numOfCarriages;
    }
    //نُكَبِّر بـ n عدد العربات في القطار
    public void IncNumOfCarriages(int n)
    {
        this.numOfCarriages = this.numOfCarriages + n;
    }
}

//-----
public class Boat extends Vehicle
{
    public Boat(string way, int maxSpeed)
    {
        super("water", way, maxSpeed)
    }
}

```

```

public class Airplane extends Vehicle
{
    private int maxHeight; // الارتفاع للطيران الاقصى
    public Airplane(int maxSpeed, int maxHeight)
    {
        super("sky", "air", maxSpeed)
        this.maxHeight = maxHeight;
    }
}
//-----
public class TransportationCompany
{
    // مصفوفة وسائل المواصلات في الشركة
    private Vehicle[] vehicles = new Vehicle[50];
    // عدد وسائل المواصلات الموجودة فعلياً
    private int counter = 0;
    public TransportationCompany()
    {
        // تُضيف وسيلة مواصلات الى مصفوفة وسائل المواصلات في الشركة
        // افترض أنه يوجد مكان لإضافة وسيلة مواصلات
    }
    public void AddVehicle(Vehicle v)
    {
        this.vehicles[counter] = v;
        this.counter++;
    }
}

```

طبق بلغة Java فئة رئيسية Program فيها عملية رئيسية, تنفذ المهمتين التاليتين :

- بناء كائن من نمط شركة وسائل مواصلات - TransportationCompany يُسمى company1 .
- إضافة قارب واحد الى الشركة company1 .
- اختر قيماً للصفات كما تشاء.

ب) في الفئة TransportationCompany عُرِّفَت العملية :

```
public void display()
{
    for (int i=0; i< this.counter ; i++)
    {
        System.out.println((i+1) + ":" + this.vehicles[i]);
    }
}
```

طبّق بلغة Java عمليات تُمكن تنفيذاً سليماً للعملية display(), بحيث تُطبع لكل وسيلة مواصلات جميع صفقاتها . عرّف العمليات بالطريقة الأكثر ملاءمة لمبادئ البرمجة الموجهة كائنات : (تغليف encapsulation، توريث inheritance، تعدد الأشكال polymorphism) .

بالنسبة لكل عملية تطبقها ، اكتب إلى أي فئة تتبع .  
لا يمكن تغيير العملية display().

ج) طبّق بلغة Java عملية , تتلقّى عدداً صحيحاً n وتضيف n عربات إلى كل القطارات التي تتبع للشركة التي لديها وسائل مواصلات من أنواع مختلفة . وثّق العملية , واكتب في أي فئة يجب تعريفها . لا يمكن تغيير العمليات الموجودة في المشروع .

**حل سؤال 13 :**

فئات تتعلق بوسائل نقل

(أ)

class Program

```

{
    static void Main(string[] args)
    {
        TransportationCompany company1 = new
        TransportationCompany();// كائن من نوع شركة //
        Vehicle boat = new Boat("SEA", 50);
        Vehicle train = new Train(100, 5);
        company1.AddVehicle(boat);// اضافة مركب //
        company1.AddVehicle(train);// اضافة قطار //
    }
}

```

بناء شركة لوسائل النقل , اضافة مركب يُبحر في البحر SEA وسرعة القصوى 50 كم/ساعة .  
 اضافة قطار وسرعة القصوى 100 كم/ساعة وبه 5 مركبات .

(ب)

دهس الدالة الوهمية toString() في الفئة الاساسية Vehicle وكل الفئات الوارثة(المشتقة).  
 الدالة تعيد نص مناسب , حسب نوع الكائن. في الفئة الاساسية Vehicle نطبق الدالة  
 toString() بالصورة التالية .

```

@Override public String toString()
{
    return " Type : " + this.type + " ; Way : " +
    this.way + " MaxSpeed : " + this.maxSpeed;
}

```

في الفئة Train نطبق الدالة toString() بالصورة التالية .

```

@Override public String toString()
{
    return "Train == >" + super.toString() + " numOfCarriages :
    " + this.numOfCarriages;
}

```

في الفئة Boat نطبق الدالة toString() بالصورة التالية

```

@Override public String toString()
{
    return "Boat == >" + super.ToString();
}

```

في الفئة Airplane نطبق الدالة toString() بالصورة التالية

```

@Override public String toString()
{
    return "Airplane == >" + super.toString() + " maxHeight : "
    + this.maxHeight;
}

```

(ج)  
نطبق دالة تتلقى عدداً صحيحاً  $n$  وتضيف  $n$  عربات إلى كل القطارات التي تتبع  
للشركة التي لديها وسائل مواصلات من أنواع مختلفة نطبقها داخل الفئة  
TransportationCompany.  
في الدالة نمشط مصفوفة وسائل النقل , ولكل القطارات نضيف لهم  $n$  من المركبات.  
فحص نوع وسيلة النقل تتم بواسطة الامر is.

```
public void AddCarriagesToTrains(int n)
{
    for (int i = 0; i < this.counter; i++)
    {
        if (this.vehicles[i] instanceof Train)
            ((Train)this.vehicles[i]).IncNumOfCarriages(n);
    }
}
```

لعرض دالة العرض Display(), وإضافة عربات لقطار الشركة ,  
بالصورة التالية . AddCarriagesToTrains

```
static void main(String[] args)
{
    TransportationCompany company1 = new
    TransportationCompany();
    Vehicle boat = new Boat("SEA", 50);
    Vehicle train = new Train(100, 5);
    Vehicle airplane = new Airplane(10000, 3000);
    company1.AddVehicle(boat);
    company1.AddVehicle(train);
    company1.AddVehicle(airplane);
    company1.Display();
    company1.AddCarriagesToTrains(2);
    Console.WriteLine();
    company1.Display();
}
```

المخرج للبرنامج :

```
C:\WINDOWS\system32\cmd.exe
1 : Boat == > Type : WATER ; Way : SEA ; MaxSpeed : 50
2 : Train == > Type : LAND ; Way : TRACKS ; MaxSpeed : 100 ; numOfCarriages : 5
3 : Airplane == > Type : AIR ; Way : SKY ; MaxSpeed : 10000 ; maxHeight : 3000

1 : Boat == > Type : WATER ; Way : SEA ; MaxSpeed : 50
2 : Train == > Type : LAND ; Way : TRACKS ; MaxSpeed : 100 ; numOfCarriages : 7
3 : Airplane == > Type : AIR ; Way : SKY ; MaxSpeed : 10000 ; maxHeight : 3000
Press any key to continue . . .
```



**سؤال 14:**

يوجد في مكتبة بلدية مجمع معلومات عن الكتب التي فيها . بالنسبة لكل كتاب يُحفظ التدرج الذي أعطاه قراء الكتاب حول مدى تمتعهم به . عندما يُعيد القارئ كتاباً إلى المكتبة يقوم بكتابة تدرجه في الحاسوب، يكتب عدداً صحيحاً بين 0 و 4 ، بحيث يشير 4 إلى أكبر مدى للتمتع. تُحفظ المعلومات حول تدرج الكتب في المجمع بحيث يكون بالإمكان معرفة عدد القراء الذين درّجوا كل كتاب بكل واحدة من الدرجات الخمس الممكنة .

توجد في مجمع المعلومات فئة تمثل الكتاب (Book) وفئة تمثل المكتبة (Library) . أمامك مخططان UML يصفان الفئتين Book و Library .

Book	
private int code	كود الكتاب
private String name	اسم الكتاب
private String genre	نوع الكتاب ( رواية، إثارة، أطفال . . . )
private int numOfCopies	عدد نسخ الكتاب الموجودة الآن في المكتبة ليست مستعارة
private int[] rating	مصفوفة إحصائية لدرجات تمتع القراء بالكتاب
....	لكل صفة معرفة العمليتان set و get
public void incNumOfCopies()	عملية تكبر ب- 1 عدد نسخ الكتاب الموجودة الآن في المكتبة (ليست مستعارة)
Public double score()	عملية تعيد علامة للكتاب ، تُحسب وفقاً لجميع درجات التمتع التي حصل عليها الكتاب .

Library	
private Book[] books	مصفوفة الكتب في المكتبة . يظهر كل كتاب مرة واحدة في المصفوفة .
	افترض أن عدد الكتب في المكتبة هو بطول المصفوفة .

عندما يُعيد قارئ معين كتاباً إلى المكتبة , يجب تنفيذ المهمات التالية :

استقبال كود الكتاب بهدف تشخيصه .

تعديل عدد نسخ الكتاب الموجودة الآن في المكتبة ( التي ليست مستعارة ) .

استقبال التدرج الذي أعطاه القارئ للكتاب، وتعديل المصفوفة الإحصائية rating للكتاب وفقاً لذلك .

طباعة بلاغ يشير إذا زادت علامة الكتاب أم انخفضت أم لم تتغير في أعقاب تدرج هذا القارئ .

طبق بلغة Java في المفئات المعطاة , العمليات المطلوبة لمعالجة إعادة كتاب إلى المكتبة . عليك تعريف العمليات بالطريقة الأكثر ملاءمة لمبنى الفئات في المشروع . بالنسبة لكل عملية اكتب الفئة التي تُعرّف فيها، وماذا تتلقى، وماذا تعيد . افترض أن لكل العمليات المكتوبة في مخطط UML يوجد تطبيق في الفئات .

ملاحظة : لا حاجة لفحص صحة المُدخلات .

**سؤال 14:**

يوجد في مكتبة بلدية مجمع معلومات عن الكتب التي فيها . بالنسبة لكل كتاب يُحفظ التدرج الذي أعطاه قراء الكتاب حول مدى تمتعهم به . عندما يُعيد القارئ كتاباً إلى المكتبة يقوم بكتابة تدرجه في الحاسوب، يكتب عدداً صحيحاً بين 0 و 4 ، بحيث يشير 4 إلى أكبر مدى للتمتع. تُحفظ المعلومات حول تدرج الكتب في المجمع بحيث يكون بالإمكان معرفة عدد القراء الذين درّجوا كل كتاب بكل واحدة من الدرجات الخمس الممكنة .

توجد في مجمع المعلومات فئة تمثل الكتاب (Book) وفئة تمثل المكتبة (Library) . أمامك مخططان UML يصفان الفئتين Book و Library .

Book	
private int code	كود الكتاب
private String name	اسم الكتاب
private String genre	نوع الكتاب ( رواية، إثارة، أطفال . . . )
private int numOfCopies	عدد نسخ الكتاب الموجودة الآن في المكتبة ليست مستعارة
private int[] rating	مصفوفة إحصائية لدرجات تمتع القراء بالكتاب
....	لكل صفة معرفة العمليتان set و get
public void incNumOfCopies()	عملية تكبر بـ 1 عدد نسخ الكتاب الموجودة الآن في المكتبة (ليست مستعارة)
Public double score()	عملية تعيد علامة للكتاب ، تُحسب وفقاً لجميع درجات التمتع التي حصل عليها الكتاب .

Library	
private Book[] books	مصفوفة الكتب في المكتبة . يظهر كل كتاب مرة واحدة في المصفوفة .
	افترض أن عدد الكتب في المكتبة هو بطول المصفوفة .

عندما يُعيد قارئ معيّن كتاباً إلى المكتبة , يجب تنفيذ المهمات التالية :

استقبال كود الكتاب بهدف تشخيصه .

تعديل عدد نسخ الكتاب الموجودة الآن في المكتبة ( التي ليست مستعارة ) .

استقبال التدرّيج الذي أعطاه القارئ للكتاب، وتعديل المصنّوفة الإحصائية rating للكتاب وفقاً لذلك .

طباعة بلاغ يشير إذا زادت علامة الكتاب أم انخفضت أم لم تتغير في أعقاب تدرّيج هذا القارئ .

طبق بلغة Java في المقتات المعطاة , العمليات المطلوبة لمعالجة إعادة كتاب إلى المكتبة . عليك تعريف العمليات بالطريقة الأكثر ملاءمة لمبنى الفئات في المشروع . بالنسبة لكل عملية اكتب الفئة التي تُعرّف فيها، وماذا تتلقّى، وماذا تعيد . افترض أن لكل العمليات المكتوبة في مخطّطي UML يوجد تطبيق في الفئات .

ملاحظة : لا حاجة لفحص صحة المُدخلات .

## حل سؤال 14:

عمليات لإعادة كتاب الى المكتبة تحوي المهام التالية :  
 استقبال رمز (رقم) الكتاب المعاد .  
 بحث عن الكتاب حسب الرقم المستقبل  
 تعديل عدد النسخ بالمكتبة  
 استقبال تدرج المتعة للكتاب من الشخص  
 تعديل مصفوفة عدادات تدرج المتعة للكتاب  
 طباعة ملاحظة تشير الى زادت علامة الكتاب أم انخفضت.

حسب ما ذكر نقوم بالعمليات والدوال الاتية :

استقبال رمز (رقم) الكتاب المعاد داخل main,  

```
System.out.println("Enter a Code of The Returned Book");
int code = input.nextInt();
```

استقبال تدرج المتعة للكتاب من الشخص داخل main,  

```
System.out.println ("Enter a Rank: [0-4]");
int rating = input.nextInt();
```

إعادة كتاب الى المكتبة :

```
public void ReturnBookToLibrary(int code, int rating)
```

ايجاد الكتاب وتعديل عدد الكتب بواحد ثم تعديل درجة المتعة من الكتاب وتطبع ملاحظة مناسبة .

إيجاد كتاب في المكتبة :

ايجاد الكتاب اولاً , معرفة داخل الفئة مكتبة Library . تتلقى كبراً ممرراً رقم الكتاب وتعيد كائن من نوع الفئة Book يصف الكتاب .

```
private Book LocateBookInLibrary(int code)
{
    int index;
    boolean found = false;
    for(index=0; (index < counter) && found != true; index++)
    {
        if (this.books[index].Code == code)
            found = true;
    }
    return this.books[index - 1];
}
```

تعديل عدد النسخ للكتاب بالمكتبة :

بواسطة الدالة IncNumOfCopies , لا حاجة للتطبيق .

تعديل مصفوفة عدادات تدرج المتعة للكتاب

تعرف بالفئة Book . تتلقى تدرج الاستمتاع بالكتاب 0-4 وتعديل المصفوفة .

```
public void UpdateRating(int rating)
{
    this.rating[rating]++;
}
```

فحص علامة الكتاب بعد التعديل:

الدرجة قبل وتفحص زادت علامة الكتاب أم انخفضت

```
public void CheckScoreStatus(double prevScore, double newScore)
{
    if (newScore > prevScore)
        System.out.println("The Books's Score Has Been Increased");

    else if (newScore < prevScore)
        System.out.println("The Books's Score Been Decreased");
    else
        System.out.println("The Books's Score Not Been Changed")
}
```

إعادة كتاب الى المكتبة :

```
public void ReturnBookToLibrary(int code, int rating)
{
    Book b = LocateBookInLibrary(code);
    b.IncNumOfCopies();
    double beforeUpdateRating = b.Score();
    b.UpdateRating(rating);
    double AfterUpdateRating = b.Score();
    b.CheckScoreStatus(beforeUpdateRating, AfterUpdateRating);
}
```

كائن بواسطة الدالة LocateBookInLibrary , تعديل عدد النسخ , تحسب درجة الكتاب تعدل مصفوفة درجات المتعة تفحص زيادة علامة الكتاب أم انخفضت

مثال :

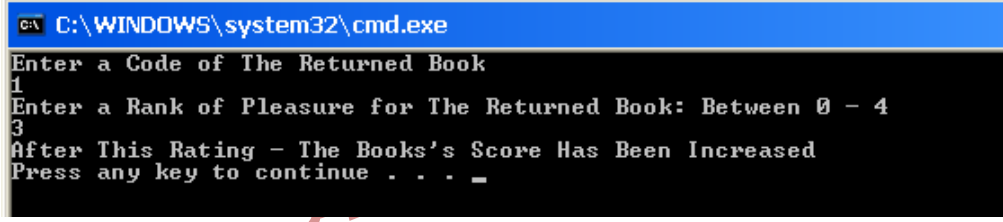
```
class Program
{
    static void main(String[] args)
    {
        Library lib = new Library();
        Book b1 = new Book(1, "Programming In Java", "Computer", 3);
        lib.AddBook(b1);

        System.out.println("Enter a Code of The Returned Book");
        int code = input.nextInt();

        System.out.println("Enter a Rank of Pleasure for The
        Returned Book: Between 0 - 4");
        int rating = input.nextInt();

        lib.ReturnBookToLibrary(code, rating);
    }
}
```

اعادة كتاب رقمه 1 الى المكتبة , درجة المتعة للكتاب 3 نتلقى المخرج :



```
C:\WINDOWS\system32\cmd.exe
Enter a Code of The Returned Book
1
Enter a Rank of Pleasure for The Returned Book: Between 0 - 4
3
After This Rating - The Books's Score Has Been Increased
Press any key to continue . . . _
```

**سؤال 15 :**

في شبكة محلات "جاستنر" 50 محلاً . تباع الشبكة المنتجات التالية :  
أجهزة MP3 , أجهزة MP4 , سماعات لاسلكية.

تحتاج الشبكة إلى برنامج حاسوب يمكن بواسطته إدارة مخزون كل واحد من المحلات .  
بالنسبة لكل واحد من المنتجات يجب على برنامج الحاسوب معالجة المعطيات التالية:

- أجهزة MP3: المنتج , الموديل , السعر , كمية المخزون , هل فيه راديو ( نعم \ لا ) , هل فيه سماعة داخلية ( نعم \ لا ) .
- أجهزة MP4: المنتج , الموديل , السعر , كمية المخزون , هل فيه راديو ( نعم \ لا ) , هل فيه سماعة داخلية ( نعم \ لا ) , طول السماعة .
- سماعات لاسلكية: المنتج , الموديل , السعر , كمية المخزون , مجال الاستقبال

كل واحد من هذه المنتجات موجود في مخزون كل واحد من محلات الشبكة , ومن كل منتج توجد موديلات مختلفة بكميات مختلفة .

المتطلبات من برنامج الحاسوب تنقسم إلى مستويين : مستوى المحل , مستوى الشبكة .

**المتطلبات من برنامج الحاسوب على مستوى المحل :**

- ✓ إعادة قيمة المخزون الذي في المحل .
- ✓ إعادة قائمة موديلات المنتج التي كميتها في المخزون المحل أصغر من عدد مطلوب limit .



## المتطلبات من برنامج الحاسوب على مستوى الشبكة:

- ✓ إعادة قيمة المخزون الذي في كل الشبكة .
- ✓ إعادة قائمة موديلات المنتج التي كميتها في المخزون في كل الشبكة أصغر من عدد مطلوب , limit .

عليك تخطيط الفئات المطلوبة لكتابة برنامج الحاسوب . يجب أن يكون تخطيط الفئات بالطريقة الأكثر ملاءمة لمبادئ البرمجة الكائنات الموجهة oop .  
(تغليف encapsulation، توريث inheritance، تعدد الأشكال polymorphism)

أ. ارسم هرمية الفئات المطلوبة . استعمل في رسمك الإشارتين التاليتين:



- ب. بالنسبة لكل فئة شملتها في الرسم، عرّف صفاتها وعملياتها .  
يجب شمل العمليات الضرورية فقط للإيفاء بالمتطلبات من برنامج الحاسوب التي وُصفت في بداية السؤال .  
بالنسبة لكل صفة، اكتب تعريفها بلغة Java وتوثيقها .  
بالنسبة لكل عملية، اكتب عنوانها بلغة Java ، واكتب توثيقاً يشمل ماذا تتلقّى وماذا تُعيد. لا حاجة لتطبيق العملية .

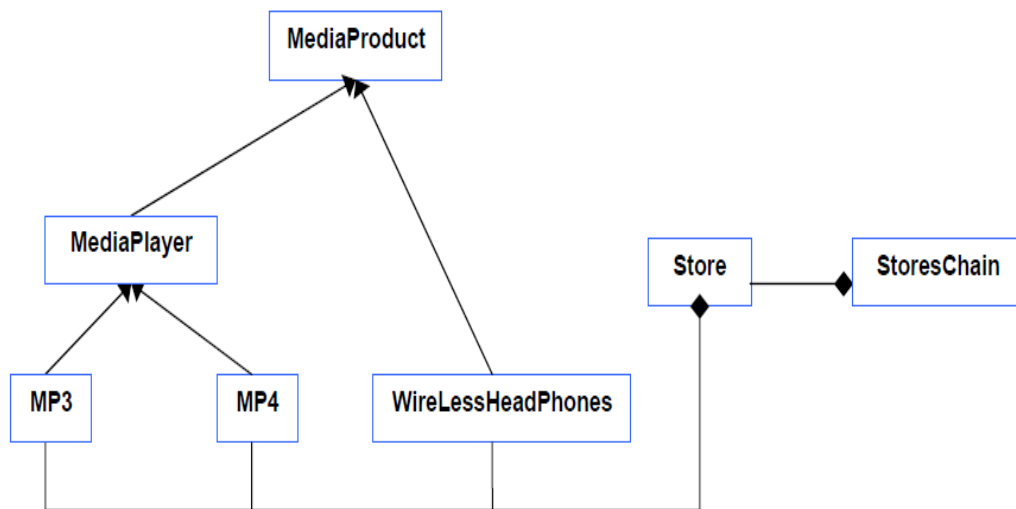
لا حاجة لكتابة عمليات بائية، وعمليات محدّدة (عمليات set) وعمليات مُعيدة (عمليات get) للصفات التي تُعرّفها .

**حل سؤال 15 :**

(أ)

الفئة	شرح عن الفئة
الفئة MediaProduct	فئة أساسية مجردة انواع من اجهزة الموسيقى والغناء
الفئة MediaPlayer	فئة مشتقة (ترث MediaProduct) تمثل اجهزة غنائية من نوع MP3 او MP4
الفئة MP3	فئة مشتقة (ترث MediaPlayer) تمثل اجهزة غنائية من نوع MP3
الفئة MP4	فئة مشتقة (ترث MediaPlayer) تمثل اجهزة غنائية من نوع MP4
الفئة WireLessHeadPhones	فئة مشتقة (ترث MediaPlayer) تمثل اجهزة غنائية من نوع سماعات لاسلكية
الفئة Store	فئة تمثل دكان , به MP3 و MP4 و سماعات لاسلكية
الفئة StoresChain	فئة تمثل محلات الشبكة

يمكن رؤية كل الفئات ترث الفئة الاساسية MediaProduct .  
 الفئة دكان تحوي كائنات من الفئات MP3, MP4 , WireLessHeadPhones .  
 فئة الدكان تحوي كائنات من الفئات الاخرى .  
 رسم تخطيطي هرمي للفئات (لإحتواء والوراثة) نصفه بالرسم التالي :



(ب)

**الفئة : MediaProduct**

<b>abstract class MediaProduct</b>	
protected String producer protected String model protected double price protected int quantity	المنتج اسم النوع (الجهاز) سعر كمية بالمخزون
... ... public double ProductStockValue()  public boolean IsProductQuantityBelowLimit(int limit)	لكل ميزة يوجد دوال-عمليات get/set للفئة تُعرّف عملية بنائية عملية تعيد سعر الاجهزة بالمخزون دالة تتلقى عدد صحيح , يمثل الحد الادنى لفحص الكمية بالمخزون , وتعيد صدق اذا كان كمية بالمخزون اصغر من العدد المتلقى . خلاف ذلك تعيد كذب

**الفئة MediaProduct**

<b>abstract class MediaPlayer : MediaProduct</b>	
protected boolean hasRadio protected boolean hasInternalSpeaker	هل يوجد مذياع (منطقي) هل يوجد سماعات (منطقي)
... ...	لكل ميزة يوجد دوال-عمليات get/set للفئة تُعرّف عملية بنائية

**الفئة MP3**

<b>class MP3 : MediaPlayer</b>	
...	للفئة تُعرّف عملية بنائية

**الفئة Store**

<b>class Store</b>	
private MP3[] MP3productModels	مصفوفة انواع الجهاز MP3
private MP4[] MP4productModels	مصفوفة انواع الجهاز MP4
private WireLessHeadPhones[] WLHPproductModels	مصفوفة انواع الجهاز سماعات لاسلكية
يمكن ان نحدد كبر المصفوفة من البداية , او نستعمل الفئة ArrayList يمكن ان يغير كبر هذه الفئة حسب عدد الكائنات.	
....	لكل ميزة يوجد دوال-عمليات get

.... .... public double StoreStockValue()	تُعرّف عملية لإضافة انواع المنتجات لمصفوفة الانواع . للفئة تُعرّف عملية بنائية عملية تعيد سعر الاجهزة بالمخزون
public ArrayList ModelsInStoreBelowLimit(string mediaType, int limit)	
عملية تتلقى نص نوع الجهاز , وعدد صحيح يمثل الحد الادنى لفحص الكمية بالمخزون وتعيد مصفوفة من نوع ArrayList من نصوص تمثل اسماء نماذج للنوع الذي كمية له بالمخزون اقل من الحد الادنى .	

**الفئة MP4**

<b>class MP4 : MediaPlayer</b>	
private double screenLength	طول الشاشة
...	لكل ميزة يوجد دوال-عمليات get/set
...	للفئة تُعرّف عملية بنائية

**الفئة StoresChain**

<b>class StoresChain</b>	
private Store[] stores	مصفوفة كائنات من الفئة التي تمثل دكان. كبير المصفوفة يكون 50 . رقم الدكان حسب كبر المصفوفة .
...	لكل ميزة يوجد دوال-عمليات get
...	تُعرّف دالة تمكن اضافة دكان الى الشبكة .
...	للفئة تُعرّف عملية بنائية
public double ChainStockValue()	دالة تعيد سعر الاجهزة بالمخزون بالشبكة
<b>public ArrayList ModelsInChainBelowLimit (string mediaType, int limit)</b>  دالة تتلقى نص نوع الجهاز وعدد صحيح يمثل الحد الادنى لفحص الكمية بالمخزون وتعيد مصفوفة من نوع ArrayList من نصوص تمثل اسماء نماذج للنوع الذي كمية له بالمخزون اقل من الحد الادنى .	
private Store BuildChainStore()	دالة مساعدة التي توحد جميع دكاكين الشبكة لدكان واحد. العملية تُستعمل كدالة مساعدة للعمليات التي تعيد سعر الاغراض بالمخزونات . واسماء الانواع التي كميتها بالمخزون اصغر من المسموح .

**الفئة WireLessHeadPhones**

<b>class WireLessHeadPhones :MediaProduct</b>	
<code>private double receptionRange</code>	مدى الاستقبال
...	لكل ميزة يوجد دوال-عمليات get/set
...	للفئة تُعرّف عملية بنائية

**مثال :**

نبنى شبكة بها دكانين بكل دكان نموذجين من الاجهزة , ولهم نفس السعر .

```

C:\WINDOWS\system32\cmd.exe
Store Stock Display:

PRODUCER      MODEL      PRICE  QUANTITY
-----
San Disk      MP3 CT015  50     10
San Disk      MP3 CT015-A 60     15
Creative      MP4 CT961  80     10
Creative      MP4 CT981  100    12
UltraSone     WLHP SR60  10      8
UltraSone     WLHP iCans 12     10

The Store Stock Value: 3600

MP3 Product's Models Below Limit 15 - In Store:
MP3 CT015

MP4 Product's Models Below Limit 13 - In Store:
MP4 CT961
MP4 CT981

WireLess HeadPhones Models Below Limit 10 - In Store:
WLHP SR60

```

```

C:\WINDOWS\system32\cmd.exe
Store Stock Display:

PRODUCER      MODEL      PRICE  QUANTITY
-----
San Disk      MP3 CT015  50     10
San Disk      MP3 CT015-A 60     15
Creative      MP4 CT961  80     10
Creative      MP4 CT981  100    12
UltraSone     WLHP SR60  10      8
UltraSone     WLHP iCans 12     10

The Store Stock Value: 3600

MP3 Product's Models Below Limit 15 - In Store:
MP3 CT015

MP4 Product's Models Below Limit 13 - In Store:
MP4 CT961
MP4 CT981

WireLess HeadPhones Models Below Limit 10 - In Store:
WLHP SR60

```

**سؤال 16 :**

أمامك مشروع فيه الفئات التالية : Base و Derived1 و Derived2 و Derived3.  
Program

```
public class Base
{
    protected int num;

    public Base(int n)
    {
        this.num = n;
    }
    protected void doSomeWork()
    {
        System.out.println("num = " + this.num);
    }
    public void run()
    {
        if (this.num % 2 == 0)
            doSomeWork();
    }
}

//-----
public class Derived1 extends Base
{
    private int num1;

    public Derived1(int n, int n1)
    {
        super(n);
        this.num1 = n1;
    }
    protected void doSomeWork()
    {
        multiplication();
    }
    public void multiplication()
    {
        super.doSomeWork();
        System.out.println("num1 = " + this.num1);
        System.out.println("num*num1=" +this.num*this.num1);
    }
}
```

```
public class Derived2 extends Base
{
    private int num2;

    public Derived2(int n, int n2)
    {
        super(n);
        this.num2 = n2;
    }
    protected void DoSomeWork()
    {
        division();
    }
    public void division()
    {
        Base.DoSomeWork();
        System.out.println("num2 = " + this.num2);
        System.out.println("num/num2=" + this.num/this.num2);
    }
}
//-----
public class Derived3 extends Base
{
    private int num3;
    public Derived3(int n, int n3)
    {
        super(n);
        this.num3 = n3;
    }
    protected void doSomeWork()
    {
        multiplication();
        division();
    }
    public void multiplication()
    {
        super.doSomeWork();
        System.out.println("num3 = " + this.num3);
        System.out.println("num*num3=" +this.num*this.num3);
    }

    public void division()
    {
        super.doSomeWork();
        System.out.println("num3 = " + this.num3);
        System.out.println("num/num3=" + this.num/this.num3);
    }
}
```

```
public class Program
{
    public static void main(String[] args)
    {
        Base[] arr = new Base[5];
        arr[0] = new Derived1(12, 22);
        arr[1] = new Derived2(33, 44);
        arr[2] = new Derived3(54, 34);
        arr[3] = new Derived1(51, 72);
        arr[4] = new Derived2(58, 99);

        for (int i = 0; i < arr.Length; i++)
            arr[i].run();

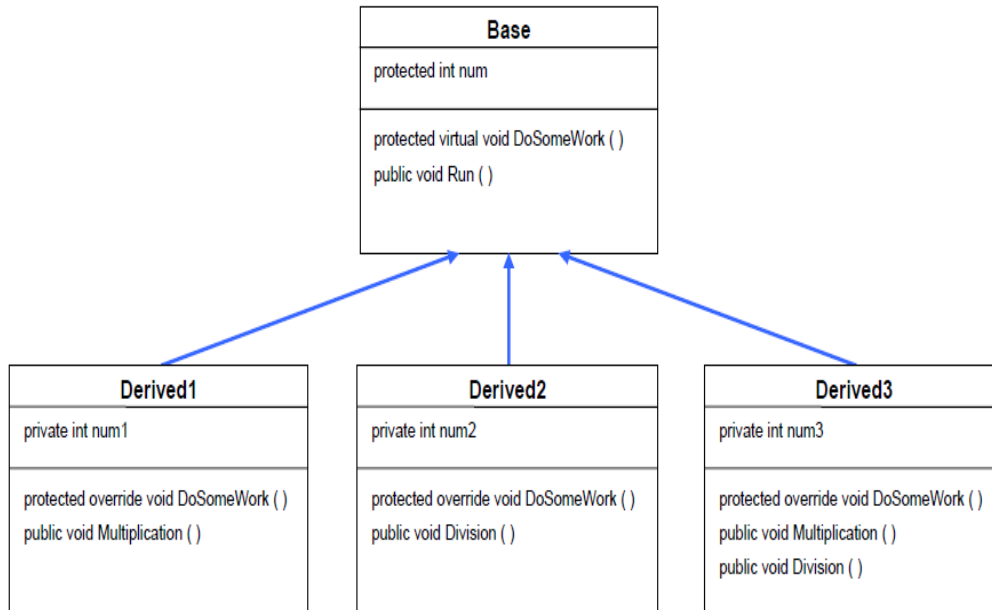
        if (arr[2] instanceof Derived3)
        {
            arr[2].run();
        }
        if (arr[3] instanceof Derived2)
        {
            arr[3].run();
        }
    }
}
```

اكتب متابعة للعملية main, في الفئة Program ، واكتب المخرَج .  
يجب أن تشمل في المتابعة قيم المتغيرات، وبالنسبة لكل كائن - قيم صفاته .



**حل سؤال 16 :**

التخطيط الهرمي للبرنامج :



الوراثة ومبدأ تعدد الاشكال , الدالة DoSomeWork() بالفئة الاساسية هي وهمية .  
وتطبيقها يدهس (override) بصورة مغايرة في كل واحدة من الفئات المشتقة .  
نُفَعِّل العملية Run() , على الكائنات في المصفوفة هذه الكائنات هي كائنات من الفئات  
المشتقة . العملية Run() تستدعي الدالة DoSomeWork ولانها دالة وهمية تفعل  
الدالة الداهسة من الفئات المشتقة .

العمليات Multiplication() / Division() من الدالة الداهسة .  
DoSomeWork() في الفئة المشتقة المناسبة حسب الكائن .

نتتبع المقطع :

a[0] - كائن من نوع Derived1 قيم ميزاته num=12, num1= 22  
a[1] - كائن من نوع Derived2 قيم ميزاته num = 33, num2= 44  
a[2] - كائن من نوع Derived3 قيم ميزاته num = 54, num2= 34  
a[3] - كائن من نوع Derived1 قيم ميزاته num = 51, num2= 72  
a[4] - كائن من نوع Derived2 قيم ميزاته num = 58, num2= 99

الموجهات-المؤشرات هي من نوع Base, لكن كل موجه يؤشر على كائنات من الفئات المشتقة.

0	1	2	3	4
Derived1	Derived2	Derived3	Derived1	Derived2

الحلقة `arr[i].Run ()`

مخرج	<code>this.num%2 == 0</code>	<code>arr [ i ]</code>			i	<code>arr.Length</code>
		نمط	num	num x		
<code>num = 12</code> <code>num1 = 22</code> <code>num * num1 = 264</code>	true	Derived1	12	num1 22	0	5
אין פלט	false	Derived2	33	num2 44	1	
<code>num = 54</code> <code>num3 = 34</code> <code>num * num3 = 1836</code> <code>num = 54</code> <code>num3 = 34</code> <code>num / num3 = 1</code>	true	Derived3	54	num3 34	2	
بدون مخرج	False	Derived1	51	num1 72	3	
<code>num = 58</code> <code>num2 = 99</code> <code>num / num2 = 0</code>	true	Derived2	58	num2 99	4	

الشرط - `if (arr [ i ] instanceof Derived)`

مخرج	<code>num % 2 == 0</code>	<code>arr [ i ] is Derived</code>	num	num x	نمط	<code>arr [ i ]</code>
<code>num = 54</code> <code>num3 = 34</code> <code>num * num3 = 1836</code> <code>num = 54</code> <code>num3 = 34</code> <code>num / num3 = 1</code>	true	<code>arr [2] is Derived3</code> true	54	num3 34	Derived3	<code>arr [2]</code>
بدون مخرج	بدون فحص	<code>arr [3] is Derived2</code> false	51	num1 72	Derived1	<code>arr [3]</code>

```

C:\WINDOWS\system32\cmd.exe
num = 12
num1 = 22
num * num1 = 264
num = 54
num3 = 34
num * num3 = 1836
num = 54
num3 = 34
num / num3 = 1
num = 58
num2 = 99
num / num2 = 0
num = 54
num3 = 34
num * num3 = 1836
num = 54
num3 = 34
num / num3 = 1
Press any key to continue . . .

```

**سؤال 17 :**

أمامك الفئة مضلع - Polygon

قسم من ميزات المضلع وعملياته موصوفة أمامك بالفئة ,

```

public class Polygon
{
    // عدد الاضلاع الأكبر
    public static int maxSides = 100;
    // مصفوفة أطوال أضلاع
    private int[] values = new int[maxSides];
    // عدد الاضلاع بالمضلع
    private int numSides = 0;

    /* عملية بنائية تعيد مضلع فارغ */
    public Polygon()
    { }

    /* عملية تعيد عدد الاضلاع بالمضلع */
    public int GetNumSides()
    {
        return this.numSides;
    }

    /* عملية تعيد مصفوفة أطوال الأضلاع للمضلع */
    public int GetValues()
    {
        return this.values;
    }

    /* دالة تعيد القيمة maxSides */
    public static void SetMaxSides(int maxSides)
    {
        Polygon.maxSides = maxSides;
    }

    /* دالة تتلقى عدد صحيح أكبر من 0 وتضيف الى المضلع ضلع بهذا الطول
    /* الفرضية: يوجد مكان بالمصفوفة للإضافة */
    public void AddSides(int x)
    {
        this.values[this.numSides] = x;
        this.numSides++;
    }

    /* دالة تستقبل من المستخدم عدد الاضلاع التي يجب اضافتها للمضلع
    /* وطول كل ضلعواضافة الاضلاع للمضلع
    /* الفرضية: يوجد مكان بالمصفوفة لإضافة الاضلاع لمصفوفة الاطوال */
    public void ReadToPolygon()
    { }
}

```

ما هي دلالة تعريف الصفة maxSides بمثابة static؟ فسّر.  
ماذا تكون دلالة تعريف الصفة maxSides لو عُرِّفَتْ بمثابة static final؟

عليك تطبيق، بلغة Java، العملية التي تضيف إلى المضلع أضلاع مضلع آخر، في كل واحدة من الحالتين i-ii اللتين أمامك :

- i. تُعرّف العملية expand بمثابة عملية داخلية في الفئة Polygon.
- ii. تُعرّف العملية expand بمثابة عملية ساكنة خارجية (ليس في الفئة Polygon)

ملاحظة : افترض أن هناك مكاناً كافياً في مصفوفة أطوال أضلاع المضلع لإضافة جميع أضلاع المضلع الآخر.

أمامك هيكل لعملية رئيسية :

```
public static void main()
{
    Polygon p1 = new Polygon();
    p1.readToPolygon();
    Polygon p2 = new Polygon();
    p2.readToPolygon();
    (*) _____
}
```

يجب أن تكتب في السطر المشار إليه بـ (\*)، أمراً يؤدي إلى إضافة أضلاع المضلع p2 إلى أضلاع المضلع p1 .

اكتب الأمر في دفترك بالنسبة لكل واحدة من الحالات i-iii :

- i. من خلال استعمال العملية الداخلية expand كما عُرِّفَتْ في الفئة polygon حسب البند جـ i .
- ii. من خلال استعمال العملية الساكنة (الستاتية) expand كما عُرِّفَتْ في البند جـ ii إذا كانت معرفّة داخل الفئة التي تتواجد فيها العملية main.
- iii. من خلال استعمال العملية الساكنة (الستاتية) expand كما عُرِّفَتْ في البند جـ iii إذا كانت معرفّة في فئة أخرى تُدعى PolygonOperations.

**حل سؤال 17 :**

أ) المتغير `maxSides` من نوع ساكن (`static`) أي فترة حياة هذه المتغيرات عند عملية التحميل للفئة للذاكرة مشتركة لجميع الكائنات من فئة معينة - `Polygons` كل `Polygon` يمكن أن يكون له 100 ضلع وغير متعلق بالكائن) يمكننا إستعمالها من غير إستنساخ للفئة أي استعمالها مباشرة بدون بناء كائن من نوع الفئة التي تتواجد بها الفئة

ب) الهدف تعريفه `final`- لا يمكن ان نغير القيمة بعد اعطائه قيمة بداية .

ج) الحل

```
public void Expand (Polygon p)
{
    if (p.numSides + this.numSides <= Polygon.maxSides)
    {
        for (int i= 0; i < p.numSides; i++)
            addSides( p.values[i]);
    }
}

public static void Expand (Polygon p1, Polygon p2)
{
    if (p1.getNumSides()+p2.getNumSides() <= Polygon.maxSides)
    {
        for (int i = 0 ; i < p2.getNumSides( ) ; i++)
            p1.addSide (p2.getValues()[i]);
    }
}
```

د)

```
i .    p1.Expand (p2);
ii.    Expand (p1,p2);
iii.   PolygonOperations.Expand (p1, p2);
```

**سؤال 18 :**

بمحل لبيع الدراجات الهوائية يمكن الدفع بوسائل الدفع التالية: نقداً, بشيك, ببطاقة اعتماد. يمكن تقسيم المبلغ الذي يُدفع بشيكات إلى عدة أقساط, وليس بالضرورة أن تكون الأقساط متساوية.

يمكن أن يتم الدفع مقابل كل شروة بوسيلة دفع واحدة أو أكثر, بحيث يفي الدمج بين وسائل الدفع بالمبلغ المطلوب.

افترض أن مبلغ الدفع وكل واحد من الأقساط هي أعداد صحيحة.

مثلاً :

أمامك عدة طرق ممكنة لدمج بين وسائل الدفع مقابل شروة بمبلغ 1000 شيقل.

- دفع كل المبلغ بوسيلة دفع واحدة : نقداً أو بشيك أو ببطاقة اعتماد.
- دفع 200 شيقل نقداً و 800 شيقل ببطاقة اعتماد.
- دفع 100 شيقل نقداً و 500 شيقل بشيكن : قيمة الأول 200 شيقل وقيمة الثاني 300 شيقل, و 400 شيقل ببطاقة اعتماد.

يحتاج المحلّ إلى برنامج حاسوب لإدارة مدفوعات المشترين.

المعلومات التي يتلقاها المحل بالنسبة لكل شروة هي :

تاريخ الشروة, ومبلغ الدفع مقابل الشروة, وتفصيل الدمج بين وسائل الدفع. بالنسبة للدفع نقداً - مبلغ الدفع.

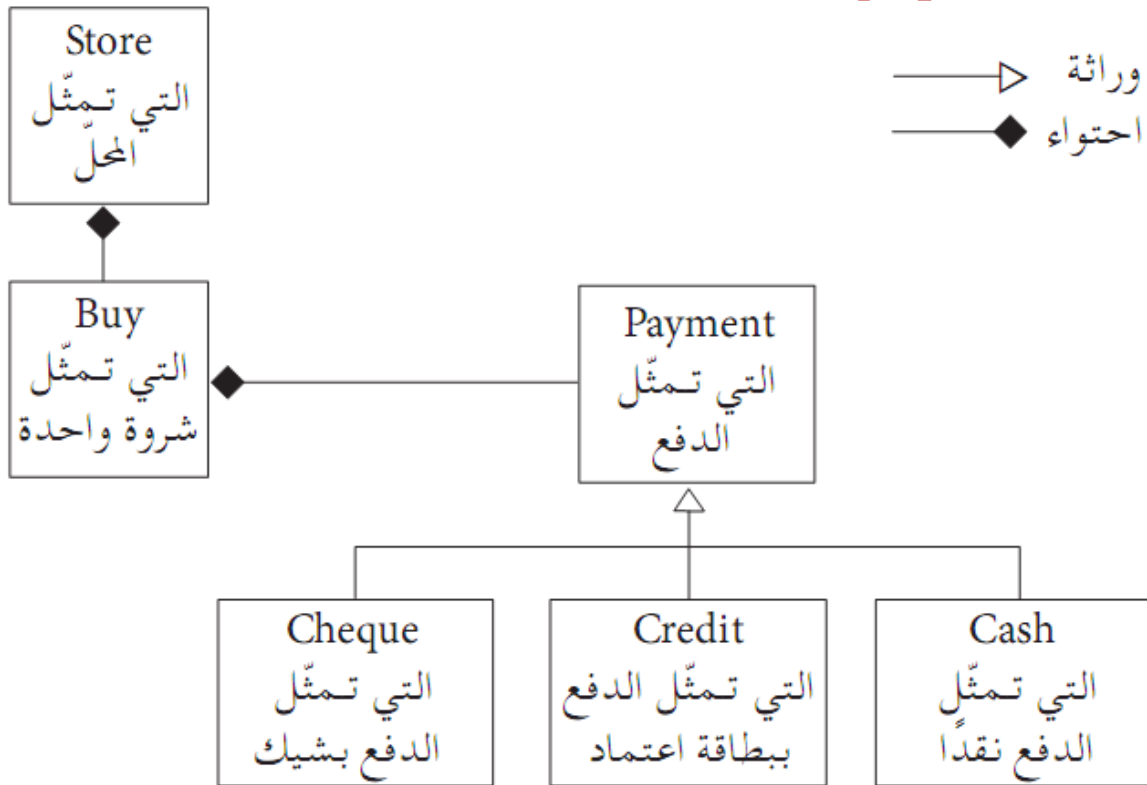
بالنسبة للدفع بشيك - مبلغ الدفع, ورقم الشيك, واسم البنك, والتاريخ المسجل على الشيك.

بالنسبة للدفع ببطاقة اعتماد - مبلغ الدفع, ورقم بطاقة الاعتماد, وتاريخ انتهاء سريان البطاقة, وتاريخ جباية المبلغ من صاحب بطاقة الاعتماد.

العمليات المطلوبة من برنامج الحاسوب هي :

- استقبال وحفظ المعلومات التي يتم تلقيها بالنسبة لكل شروء .
- فحص أنّ مجموع كل الأقساط مقابل شروء واحدة يساوي مبلغ الشروء.
- طباعة وصل للشروء.
- بالنسبة لتاريخ معين , حساب مجموع أسعار كل الدراجات الهوائية التي تم بيعها في ذلك التاريخ .

أمامك تخطيط لهرمية الفئات اللازمة لكتابة برنامج الحاسوب



بالنسبة لكل فئة في التخطيط ، عرّف صفاتها وعملياتها بالطريقة الأكثر ملائمة لمبادئ البرمجة موجهة الكائنات

(تغليف encapsulation، توريث inheritance، تعدد الأشكال polymorphism)

يجب فقط شمل العمليات الضرورية للإيفاء بالمطلوب من برنامج الحاسوب الذي وُصف في السؤال .

لا حاجة لكتابة عمليات بائية وعمليات محدّدة (عمليات set) وعمليات مُعيدة (عمليات get) .

افترض أنّ هناك فئة Date ، تمثل التاريخ .

بالنسبة لكل صفة، اكتب تعريفها بلغة Java ، واكتب توثيقها .

بالنسبة لكل عملية، اكتب عنوانها بلغة Java ، واكتب توثيقاً يشمل ماذا تتلقى وماذا تُعيد. لا حاجة لتطبيق العملية .

نسخة تجريبية - ليست للنشر



$$\{ \}$$
$$\{ \}$$

رقم بطاقة الإئتمان //  
تاريخ انتهاء صلاحية //  
تاريخ السحب //

$$\{ \}$$

رقم الشيك //  
اسم البنك //  
تاريخ الشيك //

$$\overline{\{ \}$$

المبلغ للدفع //

$$\{ \}$$

```

t total; // مبلغ المشتريات
te purchaseDate; // تاريخ الشراء
yment[ ] payments; // دفعات مختلفة
t count; // عدد الدفعات
هل ثمن جميع الدفعات مساو للمبلغ الكلي //

```

{

كل مشتريات الدكان  
عدد المشتريات

دالة تتلقى تاريخ الشراء, مبلغ الدفع وطريقة الدفع //

```
public void AddPurchase
    (Date day, int total, Payment[] payments);
```

مجموع المشتريات في يوم معين //

**سؤال 19 :**

أمامك واجهتا التطبيق : IPrintBinary, IPrintHtml  
والفئات : CreateReport, Page2, Page1, Page

```
public interface IPrintHtml
{
    public void createHlml();
}
//-----
public interface IPrintBinary
{
    public void createBinary();
}
//-----
public class Page implements IPrintBinary
{
    protected int num;

    public Page()
    { }
    public Page(int n)
    {
        this.num = n;
    }
    public void print ()
    {
        System.out.println(this.num);
    }

    public void createBinary()
    {}
}
//-----
public class Page1 extends Page
{
    protected int num1;
    public Page1 (int n, int n1)
    {
        super(n);
        this.num = n1 ;
    }

    public void print()
    {
        super.print();
        System.out.println(this.num1);
    }
}
```

```
public void createBinary()
{
    System.out.println("Binary Data: num =" +
        this.num + " num1 =" + this.num1);
}

//-----
public class Page2 extends Page implements IPrintHtml
{
    protected int num2;
    public Page2(int n, int n2)
    {
        super(n);
        this.num2 = n2;
    }
    public void print()
    {
        super.print();
        System.out.println(this.num2);
    }

    public void createHtml()
    {
        System.out.println("Html Data: num=" + this.num +
            ", num2 =" + this.num2);
    }
}

//-----
public class CreateReport
{
    public static void createBinaryDoc (IPrintBinary doc)
    {
        System.out.println("*** Binary Doc ***");
        doc.createBinary();
    }

    public static void CreateHtmlDoc(IPrintHtml doc)
    {
        System.out.println("*** Html Doc ***");
        doc.createHtml();
    }
}
```

أمامك ثلاث قطع i-iii مكتوبة بلغة Java.

بالنسبة لكل واحدة من القطع، حدّد إذا كانت قانونية أم غير قانونية. علّل تحديداتك

- i     `Page1 doc1 = new Page1 (10 ,20);`  
       `CreateReport.createHtmlDoc (doc1);`
- ii    `Page doc2 = new Page1 (30, 40);`  
       `CreateReport.createHtmlDoc(doc2);`
- iii   `IPrintBinary doc3 = new Page1 (50 ,60);`  
       `CreateReport.createBinaryDoc (doc3);`

(ب) العملية `writeHtmlDoc` , التي أمامك أضيفت إلى الفئة `CreateReport`.  
 بعد الإضافة أبلغ الكومبايلر عن خطأ . اشرح ما هو الخطأ ، وصّحه

```
public static void WriteHtmlDoc(Page doc)
{
    if (doc instanceof Page2)
        doc.createHtml();
}
```

اكتب مُخرَج قطعة البرنامج التي أمامك:

```
Page2 doc1 = new Page2 (11 , 22);
CreateReport.createHtmlDoc(doc1);
doc1.print();

Page2 doc2 = new Page1 (33 , 44);
CreateReport.createBinaryDoc(doc2);
doc2.print();

Page2 doc3 = new Page2 (55 , 66);
CreateReport.createBinaryDoc(doc3);
doc3.print();
```

**حل سؤال 19 :**

(أ)

i. غير قانوني doc1 هو كائن من نوع Page1. Page1 يرث من Page الذي هو بدوره يرث - IPrintBinary. لا يوجد وراثة من - IPrintHtml لهذا لا يمكن إرساله الى الدالة CreateReport.createHtmlDoc .

ii. غير قانوني. الفئة Page لا ترث - IPrintHtml لهذا لا يمكن ان ينجح الكومبايلير بترجمة الامر .

iii. قانوني . يحفظ كائن من نوع Page1 يرث من Page الذي يطبق IPrintBinary .  
المناداة على الدالة CreateReport.createBinaryDoc تتلقى كائن من نوع IPrintBinary .

(ب)

الخطأ يكون بالدالة - () doc.createHtml لأن للكائن Page لا يوجد الدالة createHtml لذلك نحتاج أن نحول (casting down) الى الأسفل بالصورة التالية :  
((Page2)doc2).createHtml ( )

(ج)

\*\*\* Html Doc \*\*\*

Html Data : num = 11 , num1 = 22

11

22

\*\*\* Binary Doc \*\*\*

Binary Data : num = 33 , num1 = 44

33

44

\*\*\* Binary Doc \*\*\*

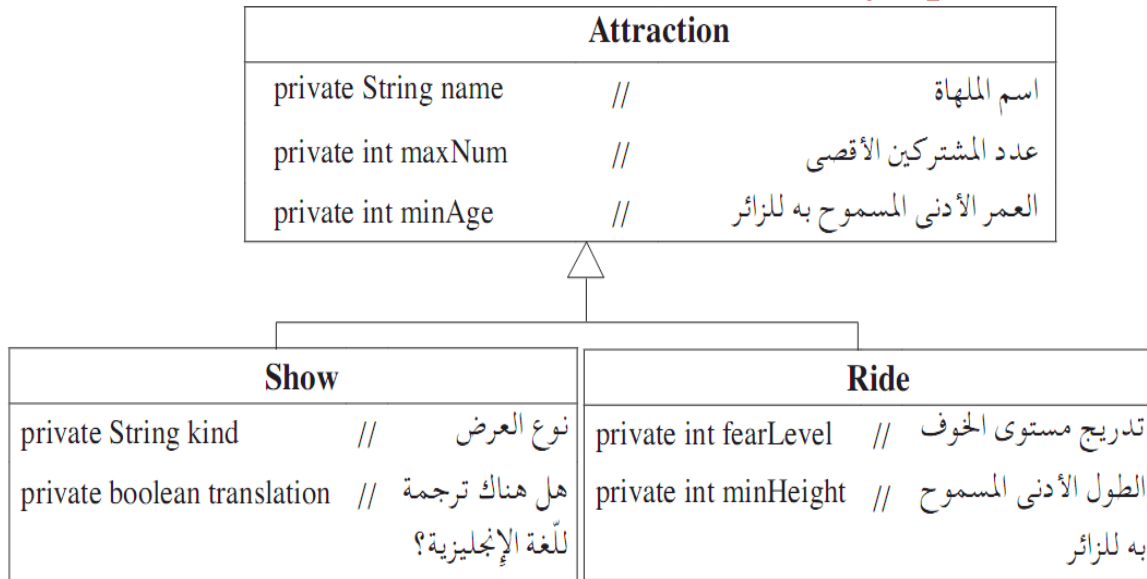
55

66

**سؤال 20 :**

توفّر مدينة ملاهٍ لزوّارها ملاهي من أنواع مختلفة . قسم من هذه الملاهي هو ألعاب (كالمرابيح أو القطار الجبلي) وقسم من الملاهي هو عروض .

معطى المخطط UML لقسم من الفئات في المنظومة المحوسّبة التابعة لمدينة الملاهي. الفئة Attraction للملهة، والفئة Ride للعبة، والفئة Show للعرض .  
الفئتان Ride و Show تَرثان من الفئة Attraction .



افترض أنه في كل واحدة من الفئات الثلاث عُرِّفت :

- عمليتا set و get لجميع الصفات.
- عملية بنائية تتلقّى بارامترات لكل صفة، وتبتدئ صفات الكائن حسب هذه البارامترات .

أ) يجب إضافة فئة Park إلى المنظومة المحوسبة وفيها الصفات:  
اسم مدينة الملاهي، مصفوفة الملاهي في مدينة الملاهي .

طبّق بلغة Java البنود i-iii في الفئة :

- i. عنوان الفئة وصفاتها.
- ii. عملية بنائية تتلقى اسم مدينة الملاهي وعدد الملاهي التي يمكن أن تحتويها مدينة الملاهي . تبتدئ العملية اسم مدينة الملاهي وتبتدئ مصفوفة الملاهي في مدينة الملاهي لتكون بالكبر الذي تمّ تلقّيه .
- iii. عملية تطبع أسماء العروض التي فيها ترجمة إلى الإنجليزية.

(ب)  
في الفئة Park معطاة العملية addAttraction التي تتلقّى ملهاة a وتضيفها إلى مصفوفة الملاهي. في مدينة الملاهي . عنوان العملية هو:

```
public void addAttraction (Attraction a)
```

اكتب ، بلغة Java ، عملية رئيسية تكون كائناً p من نمط Park اسمه lunafun فيه مكان لـ 30 ملهاة . كذلك تكون العملية الرئيسية كائنين : أحدهما من نمط Ride والآخر من نمط Show , وتضيفهما إلى مصفوفة الملاهي لـ p .  
اختر قيمياً كما تشاء لإبتداء الكائنين.

**حل سؤال 20 :**

```
class Park
{
    private string parkName;    // اسم متنزه الالعاب

    // عدد الالعاب الحالية
    private int currentNumberOfAttractions;

    private Attraction [] attractions ; // الالعاب المتوفرة

    public Park (string name, int maxAttractions)
    {
        this.parkName = name;
        attractions = new Attraction [maxAttractions];
        currentNumberOfAttractions = 0;
    }

    public void PrintShowsWithTranslation( )
    {
        for (int i=0; i < currentNumberOfAttractions; i++)
            if (attractions[i] instanceof Show)
                if ((Show)attractions[i]).getTranslation()
                    System.out.println(attractions[i]);
    }
}

ج .

public static void Main (string[] args)
{
    Park lunafun = new Park ("lunafun", 30);
    Attraction a = new Ride ("DareDevil", 10,14,5, 1);
    lunafun.Add(a);
    a = new Show ("Snow White", 100,0, "Childen", true);
    lunafun.Add(a);
}
```



**سؤال 21 :**

في حديقة الحيوانات "ZOOZOO" الحيوانات تجيد "الكلام". الحيوانات الجائعة تُصدر صوتاً طلباً للطعام. عندما يصل المربي لإطعام الحيوانات، تستقبله الحيوانات بندااءات فرح. تُصدر الحيوانات صوتاً عندما تنتهي من تناول الطعام.

أمامك مشروع يحاكي حديقة الحيوانات (ZOOZOO)، والحيوانات في حديقة الحيوانات، و "أقوال الحكمة" التي تُصدرها.

الحيوان (Animal) يمكن أن يكون زاحفاً (Reptile) أو حيواناً بحرياً (Marine). الزاحف (Reptile) يمكن أن يكون ثعباناً (Snake) أو تمساحاً (Crocodile)، الحيوان البحري (Marine) يمكن أن يكون سمكة ذهبية (GoldFish). يُحفظ لكل حيوان اسمه (name). بالنسبة للزاحف يُحفظ أيضاً طوله (len)، وبالنسبة للحيوان البحري يُحفظ أيضاً عمق غرضه الأقصى (depth).

العملية (hungry) تُعيد أقوال الحيوان الجائع. العملية (caretaker) تُعيد أقوال الحيوان الذي يستقبل مربيه. العملية (satisfied) تُعيد أقوال الحيوان الذي انتهى من تناول طعامه.

```
public class Animal
{
    private String name;           // اسم الحيوان
    public static int count = 0;    // عداد الحيوانات

    public Animal(String name)
    {
        this.name = name;
        count++;
    }

    public String hungry()
    { return this.name + "is hungry!"; } // جائع

    public String caretaker()
    { return "Yammi"; }                 // استقبال المربين

    public String satisfied()
    { return "Finish eating"; }         // إنهاء تناول الطعام
}
```

```
//-----
public class Reptile extends Animal
{
    private int len;                //الطول

    public Reptile(String name, int len)
    {
        super(name);
        this.len = len;
    }

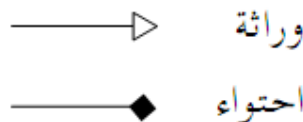
    public String caretaker()
    { return "Crawling: " + super.caretaker(); }
}
//-----
public class Snake extends Reptile
{
    public Snake(String name, int len)
    {
        super(name, len);
    }
    public String satisfied()
    {
        return "Tssss";
    }
}
//-----
public class Crocodile extends Reptile
{
    public Crocodile(int len)
    {
        super("crocki" + count, len)
    }
    public String satisfied()
    {
        return base.Satisfied() + " whaamn";
    }
}
//-----
public class Marine extends Animal
{
    private double depth;

    public Marine(String name, double depth)
    {
        super(name);
        this.depth = depth;
    }
    public String caretaker() { return "Swimming"; }
}
```

```
//-----
public class Goldfish extends Marine
{
    public Goldfish(string name)
    { super(name, 0.2) }
    public using satisfied()
    { return "Bloop bloop"; }
}
//-----
public class ZooZoo
{
    private Animal[] animals;
    public ZooZoo()
    {
        this.animals = new Animal[4];
        this.animals[0] = new Snake("snaki", 50);
        this.animals[1] = new Crocodile(78);
        this.animals[2] = new Goldfish("goldi");
        this.animals[3] = new Crocodile(103);
    }
    public void print()
    {
        for (int i = 0; i < this.animals.length; i++)
        {
            System.out.println(this.animals[i].hungry());
            System.out.println(this.animals[i].caretaker());
            System.out.println(this.animals[i].satisfied());
            System.out.println("*****");
        }
    }
}

public class Test
{
    public static void Main(String[] args)
    {
        ZooZoo zoo = new ZooZoo();
        zoo.print();
    }
}
```

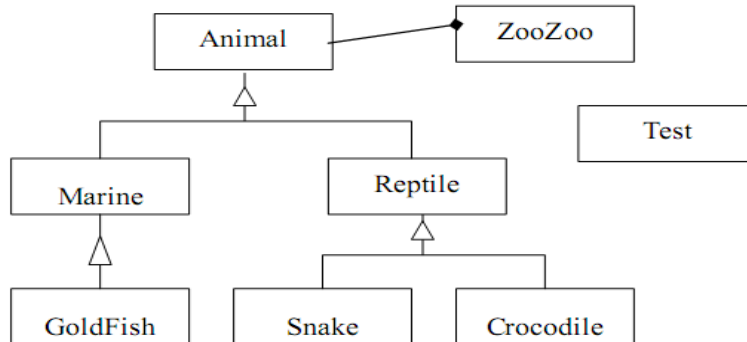
أ) ارسم هرمية الفئات المعرفة في المشروع. استعمل الإشارتين التاليتين:



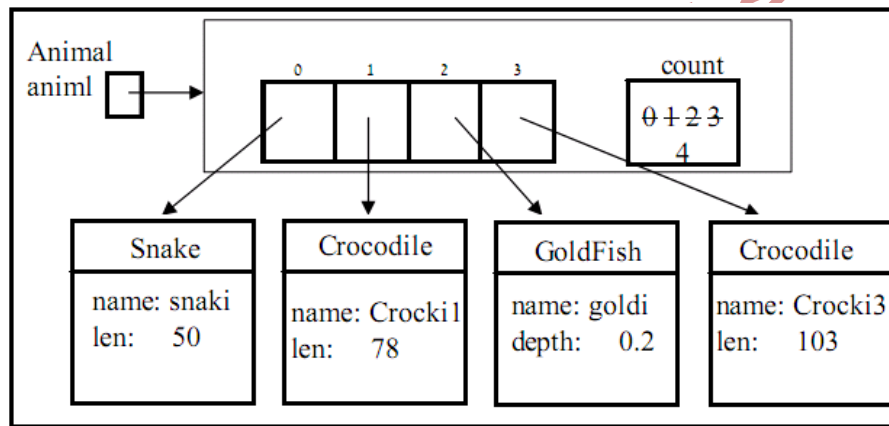
ب) اكتب متابعة للعملية main التي في الفئة Test ، واكتب المخرج .  
يجب أن تشمل المتابعة قيم المتغيرات ، وبالنسبة لكل كائن - قيم صفاته

**حل سؤال 21 :**

(أ)



(ب)



المخرج :

snaki is hungry !

Crawling: Yammi

Teeee

\*\*\*\*

Crocki1 is hungry !

Crawling: Yammi

Finish eating Whaamm

\*\*\*\*

goldi is hungry !

Swimming

Bloop bloop

\*\*\*\*

Crocki3 is hungry !

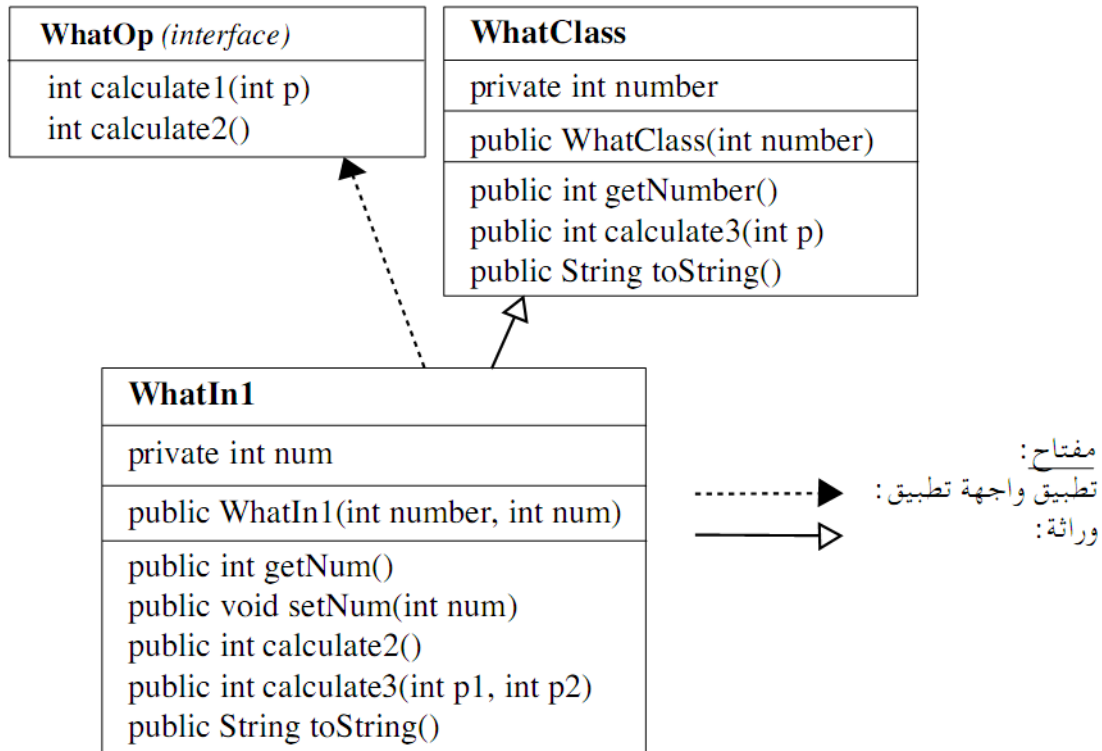
Crawling: Yammi

Finish eating Whaamm

\*\*\*\*

**سؤال 22 :**

أمامك المخطط UML الجزئي :



هل تطبيق العملية `public int calculate1(int p)` في الفئة `WhatClass` ، يفي  
 بمتطلب تطبيقها في الفئة `WhatIn1` ؟ علل إجابك .  
 تطبيق العملية البنائية في الفئة `WhatIn1` هو :

```

public WhatIn1 (int number, int num)
{
    this.number = number ;
    this.num = num ;
}
  
```

هل العملية سليمة؟ إذا كانت سليمة - صف تنفيذ العملية؟ إذا لم تكن سليمة - صحّ  
 العملية (لا تغيّر شيئاً باستثناء العملية نفسها) . اكتب العملية المصحّحة في دفترك .  
 في الفئة `WhatIn1` طُبِّقَت العملية `calculate2()` ، التي تُعيد القيمة الصحيحة لمعدّل  
 صفات الكائن .

```

Public int calculate2()
{
    return (int) ( (this.number + this.num) / 2 );
}
  
```

هل العملية سليمة؟ إذا كانت سليمة - صف تنفيذ العملية, إذا لم تكن سليمة - صحح العملية (لا تغيّر باستثناء العملية نفسها). اكتب العملية المصححة في دفترك.  
أمامك وصف جزئي للفئة WhatIn2:

Class WhatIn2 extends WhatIn1
private int sum
public WhatIn2(int number, int num, int sum)
public int Calculate3(int p1, int p2, int p3)

(1) هل يمكن الاعتماد على العملية البائية الاختيار الافتراضي (default) بدلاً من تعريف عملية بائية في الفئة WhatIn2؟ علل إجابك.

(2) العملية calculate3 طُبِّقَت في الفئات الثلاث WhatClass و WhatIn1 و WhatIn2 على النحو الآتي:

class WhatClass	public int claculate3(int p) { return this.number * p ; }
class WhatIn1	public int claculate3(int p1, int p2) { return this.claculate3(p1) + this.num * p2 * p2 ; }
class WhatIn2	public int claculate3(int p1, int p2) { return this.claculate3(p1, p2)+this.sum *p3*p3*p3 ; }

افترض أن الكائن obj هو من نمط WhatIn2 , وقيم صفاته هي :

number = 1  
num = 2  
sum = 3

أمامك أمر مكتوب في العملية الرئيسية :

System.out.println(obj.calculate3(1000, 100, 10));

بيّن متابعة تنفيذ الأمر. تطرّق في المتابعة إلى استدعاء العمليات وإلى قيم صفات الكائن. اكتب المخرج الذي ينتج.

**حل سؤال 22 :**

(أ) نعم

WhatIn1 ترث- WhatClass لذلك جميع الدوال والعمليات داخل WhatClass هن جزء من الدوال والميزات للفئة WhatIn1 , لذلك هي عليها ان تطبق الدوال والعمليات المعروفة داخل الواجهة .  
أي أن- WhatIn1 تتصرف مثل- WhatOp .

(ب) العملية غير سليمة , الفئة وارثة من WhatClass لذلك يجب عليها ان تفعل العملية البنائية للفئة الاساسية (الموروثة) . بالصورة :

```
public WhatIn1(int number, int num)
{
    super(number);
    this.num = num;
}
```

(ج) العملية غير سليمة

الميزة number هي private , اذا لا يسمح باستعمالها الا في عمليات داخلية للفئة أي انها مخفية على الفئات الاخرى .  
التصحيح - استدعاء الدالة () getNumber :

```
public int calculate2()
{
    return (int)((this.getNumber() + this.num)/2);
}
```

(د)

(1) لا يمكن الاعتماد على عملية بنائية الافتراضية . من اللحظة التي بُنيت عمليات بنائية في الفئة الاساسية WhatIn2 , ابطلت الامكانية لعملية بنائية افتراضية .

اذا حذفنا العمليات البنائية للفئات WhatClass و WhatIn1 تشغل العملية البنائية الافتراضية .

(2)

```
Console.WriteLine ( obj.calculate3 (1000, 100, 10) ) ;
```

p1	p2	p3	calculate3( p1,p2, p3)		
1000	100	10	$\frac{21000}{24000} + 3*10*10*10$	calculate3(p1, p2)	
				$\frac{1000}{21000} + 2*100*100$	Calculate3(p1)
				$1*1000$	
				1000	

المخرج : 24000

كل العمليات معروفة في الفئة WhatIn2 التي ترث كل الفئات الاخرى.

**سؤال 23 :**

أمامك عدة مبادئ للبرمجة موجهة الكائنات :

- تغليب (تغليب) - encapsulation
- تحميل زائد - overloading
- وراثة - inheritance
- دَهِس - overriding
- تعدد الأشكال - polymorphism

سيكون في هذا السؤال تطرّق إلى جزء منها .

أمامك تطبيق جزئي للفئتين: Davar , Stam .

```
public class Stam
{
    private char x;

    public Stam() { this.x = '*' ; }
    public Stam(char c) { this.x = c; }
    public Stam getStam() { return this; }
    public string toString() {return "x=" + this.x;}

    public boolean isSame1(Stam other)
    {   تطرق الى التطبيق لاحقاً في السؤال //   }

    public bool isSame2(Stam other)
    {   تطرق الى التطبيق لاحقاً في السؤال //   }

    public void same (Stam other)
    {
        if ( this.isSame1 (other))
            System.out.println(this + " same1 as " + other);
        else
            System.out.println(this + " not same1 as " + other);

        if ( this.isSame2(other))
            System.out.println(this + " same2 as " + other);
        else
            System.out.println(this + " not same2 as " + other);
    }
    public void print()
    { System.out.println( this.toString()); }

    public void print (Stam other) { this.same(other) ; }
}
```



```

public class Davar extends Stam
{
    private int y;

    public Davar ()
    { super(); this.y = 0; }
    public Davar (char c)
    { super(c); this.y = 0; }
    public Davar (char c , int num)
    { super(c); this.y = num; }
    public string toString()
    {return "Davar: " + base.ToString(); }
}

```

أ) عُرِّفت في الفئة Stam عمليتان بنائيتان .

(1) ما هو مبدأ البرمجة موجهة الكائنات (من بين المبادئ التي ذكرت في بداية السؤال) الذي يمكن ذلك؟

(2) كيف يختار الكومبيلر أية عملية بائية يجب تفعيلها؟

ب) أي مبدأ للبرمجة موجهة الكائنات (من بين المبادئ التي ذكرت في بداية السؤال) مطبق في العملية toString() في الفئة Davar؟ علّل

ج) معطاه الفئة الرئيسية التالية :

```

public class Program
{
    public static void main(String[] args)
    {
        Stam[] s = new Stam[6] ;
        s[0] = new Stam();
        s[1] = new Davar();
        s[2] = new Stam('b');
        s[3] = new Davar('b');
        s[4] = new Davar('a' , 0);
        s[5] = s[2].getStam();

        for (int i = 0; i < s.Length; i++)
        {
            s[i].print();
        }

        s[1].print(s[0]);
        s[2].print(s[5]);
        s[3].print(s[4]);
    }
}

```

القطعة 1

القطعة 2

القطعة 3

1) اعرض المصفوفة التي تُبنى في القطعة 1 (في الفئة الرئيسية المعطاة). بالنسبة لكل واحد من الكائنات، اكتب قيم صفاته .

2) اكتب مخرج الحلقة في القطعة 2 .

3) أي مبدأ للبرمجة الكائنات الموجهة (من بين المبادئ التي ذكرت في بداية السؤال) ينعكس في القطعة 2 ؟

4) طبق العمليتين isSame1() , isSame2() اللتين في الفئة Stam بحيث يكون المخرج الذي ينتج من القطعة 3 ، المخرج التالي:

```
Davar: x = * same1 as x = *  
Davar: x = * not same2 as x = *  
x = b same1 as x = b  
x = b same2 as x = b  
Davar: x = b not same1 as Davar: x = a  
Davar: x = b not same2 as Duvar: x = a
```

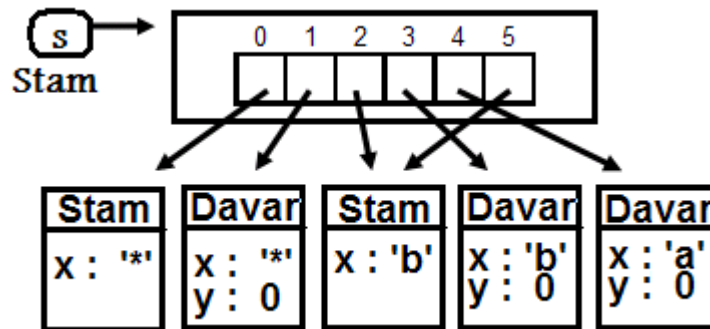
**حل سؤال 23 :**

(أ)

- (1) مبدأ البرمجة موجهة الكائنات هو دَهِس - overloading  
 (2) البرنامج يختار العملية المناسبة حسب عدد البارامترات التي تتلقاها العملية او نوع البارامترات.

(ب) العملية toString تُعرّف من جديد (دهس overriding) العملية التي وُرِثت . داخل العملية يوجد استدعاء للعملية التي ورثت (inheritance) .

(1)



(2)

مخرج الحلقة في المقطع 2 :	هذا العامود لا يطبع
s [ 0 ] :	x = *
s [ 1 ] :	Davar: x = *
s [ 2 ] :	x = b
s [ 3 ] :	Davar: x = b
s [ 4 ] :	Davar: x = a
s [ 5 ] :	Xx = b

- (3) تعدد الأشكال – مصفوفة من نوع الفئة الاساسية يمكنه أن يؤشر على كائنات من الفئات المشتقة .  
 كائن من الفئة المشتقة يمكنه أن يُفَعِّل دالة او عملية ورثها والموجودة في الفئة الموروثة مثل الدالة print().

(4)

```
public boolean isStam1 (Stam other)
{
    return this.x == other.x ;
}
public boolean isStam2 (Stam other)
{
    return this.equals( other );
}
```

**سؤال 24 :**

في مصنع لإنتاج الشموع يوجد مكان لـ 12 خط إنتاج على الأكثر. يُنتجون في كل خط إنتاج طرازاً واحداً من الشموع . توجد لكل طراز عدة ألوان يمكن إنتاج الشموع منها في هذا الطراز .

كل شمعة هي بلون واحد فقط من بين الألوان التي في طرازها.

بالنسبة لكل طراز شموع تحفظ المعلومات : اسم الطراز، وكود الطراز، وأسماء الألوان التي ينتج منها , وكمية الشموع الموجودة في المخزن من كل واحد من ألوان الطراز، وكمية الشموع القصوى التي يمكن تخزينها من هذا الطراز (من كل الألوان معاً).

مثال لطراز شموع :

- ❖ الاسم : "سوسنة" .
- ❖ الكود : 11 .
- ❖ أسماء الألوان التي يُنتج منها : أحمر , أبيض , أصفر .
- ❖ كمية الشموع في المخزن : 200 شمعة حمراء و 350 شمعة بيضاء و 70 شمعة صفراء .
- ❖ الكمية القصوى للشموع من هذا الطراز (من كل الألوان معاً) التي يمكن تخزينها: 100000 .

تُحفظ في المصنع معلومات بالنسبة لكل طراز شموع يُنتج فيه.

- كل مرة يريدون فيها إنتاج شمعة من طراز معين , يجب التطرق إلى الشرطين التاليين :
- يتم اختيار لون الشمعة حسب كمية الشموع المخزونة : اللون الذي كمية الشموع التي منه في المخزن هي الأقل من نفس الطراز ، هو اللون الذي يتم اختياره لإنتاج الشمعة (إذا كان هناك أكثر من لون واحد كهذا يتم اختيار أحدها) .
  - كمية الإنتاج تخضع لاختيار المشغل , لكن لا يمكن أن تزيد كمية الشموع الكلية من طراز معين (الشموع التي تُنتج والشموع المخزونة) عن الكمية القصوى من هذا الطراز التي يمكن تخزينها .

بنى مهندس البرمجة في المصنع مشروعاً فيه ثلاث فئات:

CandleKind	فئة تمثل طراز الشموع
Factory	فئة تمثل المصنع
Run	فئة رئيسية

### أمامك أجزاء من الفئة CandleKind :

```
public class CandleKind
{
    private String name ; // اسم الطراز
    private int code ; // كود الطراز
    private String [ ] colcors ; // مصفوفة ألوان الطراز المتنوعة
    private int [ ] amounts; // كمية كل لون في المخزن بالتلاؤم
    private int maxTotalAmount ; // الكمية القصوى للخرن

    public CandleKind( String name, int code, String [ ]
    colors, int maxTotalAmount)
    {
        this.name = name ;
        this.code = code ;
        this.corcors = colors ;
        this.amounts = new int [ colors.Length ] ;
        for ( int i=0 ; i< amount.length; i++)
        {
```

```

this.amounts[i] = 0;
}
this.maxTotalAmount = maxTotalAmount;
}
public void Update( String color, int amount)

العملية تزيد بكمية amount كمية الشموع بالون color التي في
المخزن//

```

أ) اكتب أمراً / أوامر لبناء كائن باسم kind1 من الفئة CandleKind بالنسبة لطرز جديد للشموع الذي يبدأون في إنتاجه في المصنع . يُنتج الطراز الجديد بثلاثة ألوان مختلفة .

اختر قيماً للصفات كما تشاء . اكتب قيم صفات الكائن بعد بنائه .

ب) اكتب عنوان الفئة Factory وصفاتها / صفاتها . اكتب توثيقاً لكل صفة .

ج) أمامك جدول يتضمن توثيقاً للعمليات 1-3 التي يجب إضافتها إلى المشروع.

اكتب لكل عملية:

i. في أية فئة (من بين الفئات الثلاث) من الملائم تعريفها . اشرح اختيارك.

ii. عنوان العملية.

1	عملية تُعيد لون الشمعة من طراز شموع معين يبدأون في إنتاجه .
2	عملية تُعيد كمية الشموع التي يمكن إنتاجها من طراز شموع معين.
3	عملية تُعيد كود طراز الشمعة الذي بالنسبة له الفرق بين الكمية التي توجد من هذا الطراز في المخزن وبين الكمية التي يمكن تخزينها هو الأكبر . إذا وُجدت عدة طرازات شموع كهذه , يعاد أحدها . إذا كانت كمية كل واحد من طرازات الشموع التي في المخزن هي الكمية القصوى التي يمكن تخزينها، يعاد الكود 999 (كود غير موجود)

(د) أمامك عملية رئيسية تُطبّق في الفئة Run . العملية تُدير عملية الإنتاج . أكمل قِطْع الكود المرقّمة بالأرقام (1)-(4) حسب تفصيل المتطلبات الذي يظهر بعد الكود .

افترض أنّ العمليات التي عرّفناها في البند "ج" موجودة، وأنّ العمليتين get ، set معرفتان لكلّ صفة في الفئة CandleKind المعطاة وفي الفئة Factory التي كتبناها في البند "ب" . إذا عرّفت عمليات إضافية، اكتب في أيّة فئة عرّفناها وطبّقها بصورة كاملة .

```
static void Main(String[] args)
{
    Factory fty;
    // من هنا فصاعداً افترض أن الكائن fly موجود مع قيم
    // الصفات التي تصف حالة المخزون في المصنع
    int code = _____(1)_____
    while( code != 999 )
    {
        CandleKind ck;
        _____(2)_____
        String colorToProduce ;
        _____(3)_____
        int amount ;
        _____(4)_____
        ck.update(colorToProduce, amount);
        code = _____(1)_____
    }
}
```

تفصيل المتطلبات :

- (1) استدعاء العملية 3 من الجدول الذي في البند "ج" .
- (2) قطعة كود تؤدي إلى احتواء المتغيّر ck طراز الشمعة الذي كوده هو code .
- (3) قطعة كود تؤدي إلى احتواء المتغيّر colorToProduce اللون الذي يجب إنتاجه من طراز الشمعة الذي كوده هو code .
- (4) قطعة كود تؤدي إلى احتواء المتغيّر amount كمية الشموع التي يمكن إنتاجها من طراز الشمعة الذي كوده هو code .

**حل سؤال 24 :**

أ) مقطع البرنامج لبناء كائن بإسم kind1 من نوع CandleKind :  
الطراز: Rose  
اللون: مصفوفة colors1 تحوي كل الألوان: أحمر , أصفر , أخضر .  
رقم الطراز: 500800  
كمية قصوى: 2010

```
String [ ] colors1 = {"red", "yellow", "green"};
CandleKind kind1 = new CandleKind ("Rose", 500800, colors1, 2010);
```

ب) عنوان وميزات الفئة Factory :  
 prodLine - مصفوفة خطوط الانتاج .  
 maxLine - عدد الاقصى لخطوط الانتاج .  
 current - عدد خطوط الانتاج الحالي .  
 (في البرنامج دائما نفرض :  $current < maxLine$ )

```
public class Factory
{
    private CandleKind[] prodLine;
    public static int maxLine = 12;
    private int current;

    public Factory()
    {
        this.prodLine = new CandleKind[maxLine];
        this.current = 0;
    }
}
```

ج) دوال وعمليات تضاف الى الفئة CandleKind :

```
// قسم ج-1
// عملية تعيد لون الشمعة من طراز معين سيبدأون بإنتاجها
public string startColorProduction { ... }
```

```
// قسم ج-2
// عملية تعيد كمية الشمع الاقصى
// الذي يمكن انتاجه من الطراز الحالي
public int posibleProductAmount() { ... }
```

عملية تضاف الى الفئة Factory :

```
// قسم ج-3
// عملية تعيد رقم الشمعة من طراز معين سيبدأون بإنتاجها
// الشمعة الموجودة بأقل كمية بالمخزن والتي يمكن انتاجها
// اذا لا نستطيع انتاج هذا النوع تعيد الدالة 999
public int getCodeMinimumAmounts () { ... }
```



(د)

مقطع البرنامج : main ()

```
Factory fty = new Factory();
```

```
int code = fty.getCodeMinimumAmounts();           //(1)
```

```
while (code != 999)
```

```
{
```

```
    CandleKind ck = fty.getProdLine(code);           //(2)
```

```
    String colorToProduce = ck.startColorProduction(); //(3)
```

```
    int amount = ck.posibileProductAmount();         //(4)
```

```
    ck.update(colorToProduce, amount);
```

```
    code = fty.getCodeMinimumAmounts();             //(1)
```

```
}
```

نسخة تجريبية - ليست للنشر

**سؤال 25 :**

أمامك الفئتان AA و BB :

```

class AA
{
    private String st;

    public AA() { this.st = "excellent"; }
    public AA(String st) { this.st = st; }
    public String GetSt() { return this.st; }
    public void SetSt(String st) { this.st = st; }
    public string ToString() { return "st=" + this.st; }
}

class BB extends AA
{
    private int num;

    public BB { super(); this.num = 1; }
    public BB(int num, String st) { super(st); this.num = Math.Abs(num); }

    public int GetNum() { return this.num; }
    public void SetNum(int num) { this.num = num; }
    public String ToString() { return super.ToString() + "num=" + this.num; }
}

```

أ) عرّف في الفئة AA عملية بوليانية باسم isLike(Object obj) ، تتلقى كائناً  
 obj من نوع Object . إذا كان الكائن obj من نمط AA وكذلك محتوى النص  
 st التابع لـ obj مطابقاً لمحتوى النص st التابع للكائن الحالي - تُعيد العملية  
 true , خلاف ذلك - تعيد false .

ب) عرّف في الفئة BB عملية تدهس (override) العملية التي عرّفها في البند "أ".  
 إذا كان الكائن obj من نمط BB وكذلك قيمة صفته num مساوية لقيمة الصفة  
 num التابعة للكائن الحالي - تعيد العملية true , خلاف ذلك - تعيد false .

(ج) أمامك قطعة من عملية رئيسية:

```
AA a = new AA("excellent");
BB b = new BB();
a = b;
if (a.IsLike(b)) System.out.println(a);
```

هل قطعة البرنامج سليمة؟  
إذا كانت سليمة - ماذا يكون مخرج القطعة؟ اكتب أية صيغة للعملية `isLike` تُفَعِّل -  
الصيغة التابعة لـ `AA` أم الصيغة التابعة لـ `BB`.  
إذا كانت غير سليمة- اشرح ما هو الخطأ ومتى سيظهر: أثناء التجميع والتحويل للغة  
الآلة أم أثناء التشغيل.

(د) أمامك قطعة من عملية رئيسية :

```
AA aa = new AA();
BB bb = new BB(2 , "excellent");
bb = aa;
if (bb.isLike(aa)) System.out.println(bb);
```

هل قطعة البرنامج سليمة؟  
إذا كانت سليمة - ماذا يكون مخرج القطعة؟ اكتب أية صيغة للعملية `isLike` تُفَعِّل -  
الصيغة التابعة لـ `AA` أم الصيغة التابعة لـ `BB`.  
إذا كانت غير سليمة - اشرح ما هو الخطأ ومتى سيظهر: أثناء التجميع والتحويل للغة  
الآلة أم أثناء التشغيل .

(هـ) اكتب عملية خارجية باسم `longString` تتلقى مصفوفة لكائنات من نمط `Object`.  
تُعيد العملية نصاً مركباً من دمج الصفة `st` لكائنات من نمط `AA` في المصفوفة ,  
على النحو التالي :

- ✓ إذا كانت للكائن الصفة `st` فقط ، النص الذي في الصفة `st` يُدمج مرة واحدة.
- ✓ إذا كانت للكائن الصفة `num` أيضاً، النص الذي في الصفة `st` يُدمج `num` مرات.
- ✓ إذا لم يكن في المصفوفة أي كائن من نمط `AA`، يُعاد نص فارغ .

**حل سؤال 25 :**

(أ) في الفئة AA :

```
public virtual bool IsLike(Object obj)
{
    return obj is AA ((AA)obj).GetSt().Equals(this.GetSt());
}
```

(ب) في الفئة BB :

```
@Override public bool IsLike(object obj)
{
    return (obj is BB) && ((BB)obj).GetNum()==this.GetNum();
}
```

(ج) مقطع البرنامج صحيح, عملية تحويل للأعلى لمتغيرات من نوع الابن (الاساس) لموجهات من نوع الاب (مشتقة) . كل موجه من نوع الاساس يمكنه ان يؤشر الى كائنات من نوع المشتقة . المخرج يكون :

```
st=excellent    num=1
```

(د) قطعة البرنامج خاطئة. لا يمكن ان نحول للأسفل (downcasting) مؤشر من نوع المشتقة أن يكون من نوع الاساس . لا يمكن لمؤشر من نوع BB أن يؤشر على كائن من نوع AA . خطأ المترجم compiler.

(هـ)

```
public static String LongString(Object[] a)
{
    String st = "";
    for (int i = 0; i < st.Length; i++)
    {
        if (a[i] is BB)
        {
            for (int j = 1; j < ((BB)a[i]).GetNum(); j++)
                st+=((BB) a[i]).GetSt();
        }
        else
            if (a[i] is AA) st+=((AA)a[i]).GetSt();
    }
    return st;
}
```

**حل آخر:** يمكن ان نفحص في البداية اذا كان من نوع AA وأيضاً ليس من BB , خلاف ذلك اذا كان AA .

**سؤال 26 :**

أمامك مشروع فيه الفئات B و D و A و OopTest .

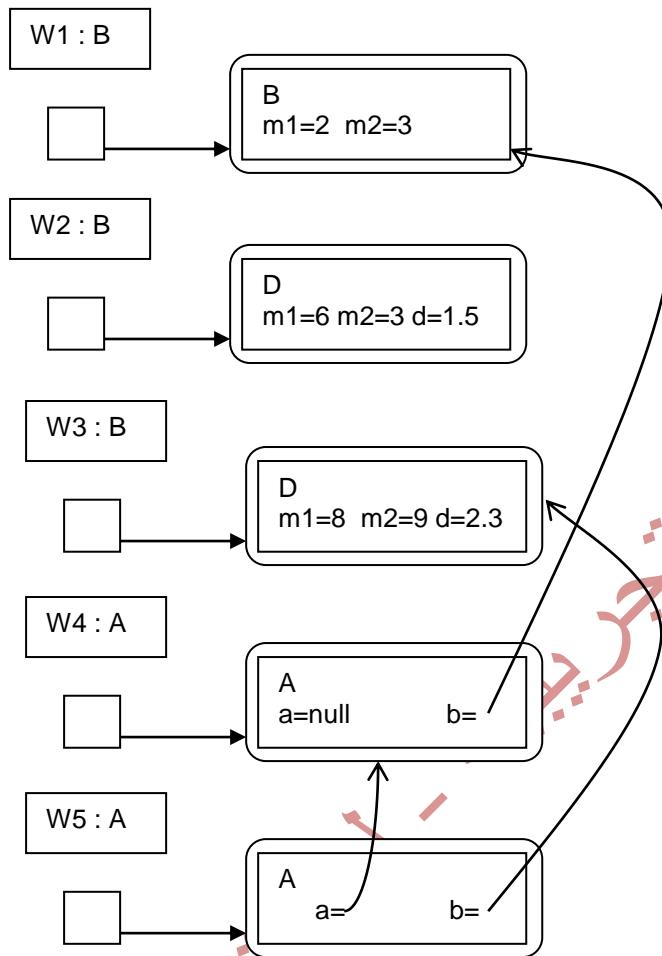
```
class B
{
    private static int numB = 0;
    private int m1;
    private int m2;
    public B(int m1, int m2)
    {
        this.m1 = m1;
        this.m2 = m2;
        numB++;
        System.out.println("B(" + m1 + ", " + m2 + "), #" + numB);
    }
}

class D extends B
{
    private static int numD = 0;
    private double d;
    public D(double d, int x)
    {
        super(x, x);
        this.d = d;
        numD++;
        System.out.println("D(" + d + ", " + x + "), #" + numD);
    }
    public D(double d, int x, int y)
    {
        super(x, x);
        this.d = d;
        numD++;
        System.out.println("D(" + d + ", " + x + ", " + y + "), #" + numD);
    }
}
```

```
class A
{
    private static int numA = 0;
    private A a;
    private B b;
    public A(A a, B b)
    {
        this.a = a;
        this.b = b;
        numA++;
        System.out.println("A Constructor, #" + numA);
    }
}

class OopTest
{
    static void main(String[] args)
    {
        B w1 = new B(2, 3);
        B w2 = new D(15, 6);
        B w3 = new D(2.3, 8, 9);
        A w4 = new A(null, w1);
        A w5 = new A(w4, w3);
    }
}
```

اكتب متابعة للعملية main في الفئة OopTest , واكتب المخرج .  
يجب أن تكتب في المتابعة قيم المتغيرات , وبالنسبة لكل كائن – قيم صفاته .

**حل سؤال 26 :**

المخرج :

B(2 , 3) , #1  
 B(6 , 6) , #2  
 D(1 , 5, 6) , #1  
 B( 8, 9) , # 3  
 D(2.3 , 8 , 9) , #2  
 A Constructor , #1  
 A Constructor , #2

**سؤال 27 :**

طوّرت شركة لتوزيع برامج الحاسوب برنامج حاسوب يتناول متواليات الأعداد الصحيحة . تم تطوير المنظومة على مراحل.

- بالنسبة لكل متوالية أعداد، يتطرقون إلى :
- (1) الحدّ الأول في المتوالية الذي رقمه التسلسلي هو 1.
  - (2) الحدّ الذي رقمه التسلسلي في المتوالية هو n.
  - (3) طباعة n الحدود الأولى في المتوالية .

طوّرت في المرحلة الأولى فئتان:  
متوالية حسابية (ASeq) - متوالية الفرق فيها بين كل حدّ والحدّ الذي قبله هو قيمة ثابتة.  
متوالية هندسية (GSeq) - متوالية حاصل القسمة (الأساس) فيها بين كل حد والحدّ الذي قبله هو قيمة ثابتة .

فيما يلي كود الفئتين اللتين طوّرتا في المرحلة الأولى:

```
public class ASeq
{
    private int first;
    private int difference;

    public ASeq(int first, int diff)
    {
        this.first = first;
        this.difference = diff;
    }

    public int TheNElement(int n)
    {
        return this.first + (n - 1) * this.difference;
    }

    public void DisplayNElements(int n)
    {
        System.out.println("The Sequence elements:");
        for (int i = 0; i < n - 1; i++)
            System.out.println(this.theNElement(i + 1) + ",");

        System.out.println(this.theNElement(n));
    }
}
```



```

public class GSeq
{
    private int first;
    private int product;

    public GSeq(int first, int product)
    {
        this.first = first;
        this.product = product;
    }

    public int TheNElement(int n)
    {
        return this.first * (int)Math.Pow(this.product, n - 1);
    }

    public void DisplyNElements(int n)
    {
        System.out.println("The Sequence elements:");
        for (int i = 0; i < n - 1; i++)
            System.out.println(this.TheNElement(i + 1) + ",");
        System.out.println(this.TheNElement(n));
    }
}

```

(أ) تتبع قطعة البرنامج التي أمامك. اعرض في المتابعة الكائن الذي يُبنى وصفاته والمُخرج .

```

ASeq aSeq = new ASeq(2, 3);
System.out.println(aSeq.TheNElement(4));
aSeq.displyNElements(5);

```

تقرّر في المرحلة الثانية من التطوير أنّه من المناسب تطوير فئة جديدة تصف متوالية ثابتة (Sequence) بحيث ترث الفئتان ASeq و GSeq من الفئة الجديدة . في المتوالية الثابتة معرفة قيمة الحدّ الأول، وجميع سائر الحدود مساوية للحدّ الأول.

(ب) أكمل تطوير المرحلة الثانية بالطريقة الأكثر ملاءمة لمبادئ البرمجة موجهة الكائنات , وحسب التعليمتين (i-ii) :

(i) طبق بشكل كامل الفئة العليا Sequence. يجب أن تنطبق الفئة إلى :

(1) الحد الأول في المتوالية الذي رقمه التسلسلي هو 1.

(2) الحد الذي رقمه التسلسلي في المتوالية هو n.

(3) طباعة n الحدود الأولى في المتوالية.

(ii) طبق الفئة ASeq من جديد بحيث ترث من الفئة Sequence.

تقرر في المرحلة الثالثة من التطوير توسيع المشروع الذي يشمل ثلاث الفئات التي تم تطويرها في المرحلة الثانية (Sequence, ASeq, GSeq) , بحيث يكون بالإمكان بالنسبة لكل متوالية , تفعيل عملية تحسب وتعيد مجموع n الحدود الأولى في المتوالية.

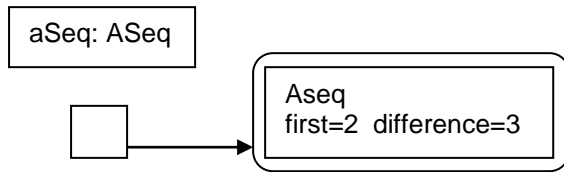
افترض أن الفئة GSeq قد تم تطبيقها من جديد بحيث ترث من الفئة Sequence.

(ج) بالنسبة لكل واحدة من الفئات Sequence, ASeq, GSeq , اكتب هل يجب إجراء تغييرات فيها بحيث يفي المشروع بمتطلبات التطوير في المرحلة الثالثة بالطريقة الأكثر ملاءمة لمبادئ البرمجة الموجهة كائنات.

إذا كان يجب إجراء تغييرات - فصلها وطبقها .

**حل سؤال 27 :**

(أ)

المخرج:

11

The sequence elements 2 , 5, 8, 11 ,14

(ب)

```

public class Sequence
{
    protected int first;

    public Sequence(int first) {    this.first = first;  }
    public virtual int TheNElement(int n)
    {    return this.first;  }
    public virtual void DisplayNElement(int n)
    {
        System.out.print("The sequence elements");
        for (int i = 0; i < n; i++)
            System.out.print(this.first + ",");
    }
}
  
```

(2)

```

public class ASeq extends Sequence
{
    private int difference;

    public ASeq(int first, int difference)
    {
        super(first);
        this.difference = difference;
    }
    @Override public int TheNElement(int n)
    {    return this.first + (n - 1) * this.difference;  }

    @Override public void DisplayNElement(int n)
    {
        Console.WriteLine("The sequence elements");
        for (int i = 0; i < n - 1; i++)
            System.out.print(this.TheNElement(i + 1) + ",");
        System.out.println(this.TheNElement(n));
    }
}
  
```

(ج) لا حاجة لأي تغيير , العملية تكتب داخل Sequence بالصورة :

```
public int SumSeq(int n)
{
    int sum = 0;
    for (int i = 1; i <= n; i++)
        sum += TheNElement(i);
    return sum;
}
```

بالفتتين الوارثتين لا حاجة بكتابة مرة اخرى , كل واحدة تحسب القيمة العددية للحد حسب العملية المكتوبه بها , لان هنا مبدأ الدهس (overriding) يفعل .

(د)

```
public static char Bigger(int n, ASeq a, GSeq g)
{
    int s1 = a.SumSeq(n);
    int s2 = g.SumSeq(n);
    if (s1 > s2)    return 'A';
    if (s2 > s1)    return 'G';
    return 'E';
}
```

**سؤال 28 :**

تركز عيادة بيطرية للحيوانات البيئية معلومات تتعلق بالأطباء ، البيطريين الذين يعملون في العيادة وبالحيوانات البيئية التي تتلقى العلاج فيها . عدد الأطباء البيطريين الذين يعملون في العيادة هو 10 (على الأكثر) وعدد الحيوانات البيئية التي تعالج في العيادة هو 500 على الأكثر.

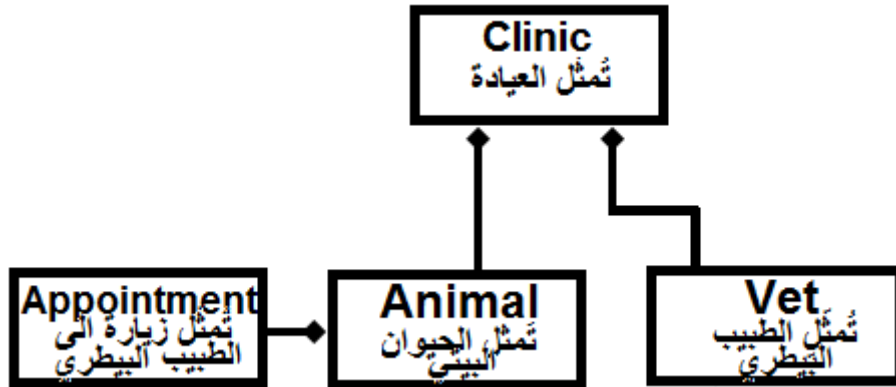
النقاط الموجهة في معالجة المعلومات هي :

- يُحفظ لكل طبيب بيطري : رقم الهوية والاسم والأقدمية في العمل بالسنوات .
- يُحفظ لكل حيوان بيتي : رقم الرخصة والاسم والنوع (مثلاً: كلب أو قطة أو أرنب) , والعمر , وتفصيل حتى 50 الزيارات الأخيرة في العيادة وعدد الزيارات المحفوظة.
- يُحفظ لكل زيارة إلى العيادة: رقم هوية الطبيب البيطري الذي عالج الحيوان البيتي في هذه الزيارة , ونص من رموز يحوي أكواد العلاجات التي حصل عليها الحيوان البيتي في تلك الزيارة . كود العلاج هو حرف كبير من بين جميع أحرف الأبجدية ABC.

جزء من العمليات التي تستطيع المنظومة تنفيذها هي:

- إعداد تقرير عن جميع الحيوانات البيئية من نوع معين (مثلاً جميع الأرانب) .
- يشمل التقرير بالنسبة لكل حيوان رقم رخصته واسمه وعمره .
- في بداية كل سنة : تعديل عمر كل حيوان بيتي , أي تكبير العمر بـ 1 , وتعديل أقدمية كل طبيب بيطري، أي تكبير الأقدمية بـ 1 .
- إعادة اسم الطبيب البيطري حسب رقم هويته .

أمامك مخطط لهرمية الفئات اللازمة لمعالجة المعلومات في العيادة:



(أ)

بالنسبة لكل فئة في المخطط ، عرّف صفاتها وعملياتها .  
عليك فقط تعريف العمليات اللازمة للإيفاء بالمتطلبات التي وصفت في مقدمة السؤال  
(التي في صفحة 44) وفي المخطط .

افتراض أنّه معطاة عمليات بنائية تتلقى بارامتراً لكل صفة , وعمليات مُعيدة  
(عمليات get), ولا حاجة لكتابتها . عمليات محدّدة (عمليات set) غير معطاة .  
بالنسبة لكل صفة - اكتب تعريفها بلغة Java واكتب توثيقها.  
بالنسبة لكل عملية - اكتب عنوانها بلغة Java , واكتب توثيقاً يشمل ماذا تتلقى وماذا تعيد.  
لا حاجة لتطبيق العملية .

(ب)

افترض أن المنظومة طُوِّرت حسب التخطيط الذي عرضته في البند أ , وأن جميع العمليات المعطاة وتلك التي عرفتھا قد تم تطبيقھا.

في الفئة clinic نضيف العملية:

```
Public void addAppointment(Animal p, String t, Vet v)
```

التي تتلقى حيواناً بيتياً p, ونصاً t لأكواد العلاجات التي حصل عليها الحيوان البيتي في الزيارة الحالية، والطبيب البيطري v الذي عالجه . تضيف العملية الزيارة إلى الحيوان البيتي .  
طبّق العملية بشكل كامل .

افترض أن عدد الزيارات السابقة للحيوان في العيادة هو أصغر من 50 .  
إذا استعملت عمليات أخرى بالإضافة إلى العمليات المعطاة والعمليات التي عرفتھا في البند أ , عليك أن تطبقھا بشكل كامل وأن تذكر بالنسبة لكل عملية في أية فئة يجب تطبيقھا .

**حل سؤال 28 :**

```
public class Vet
{
    private int id;
    private String name;
    private int vetek;

    public void SetVetek() // تعديل سنوات العمل للطبيب البيطري ب 1
    {
        this.vetek++;
    }
}
```

```
public class Animal
{
    private int numR;
    private String name;
    private String sug;
    private int age;
    private int numVet;
    private Appointment[] lastVisits;

    public void SetAge() // تعديل عمر الحيوان بواحد
    {
        this.age++;
    }

    public Appointment[] GetLastVisits()
    {
        return this.lastVisits;
    }
}
```

```
public class Appointment
{
    private String kodes;
    private int idVet;
}
```



```
public class Clinic
{
    private Vet [] veterinarians;
    private Animal [] animals;

    // الدالة تتلقى نوع حيوان وتطبع تقرير كل الحيوانات من هذا النوع
    public void AllAnimalsSameKind(string kind)
    {
        int i = 0;
        while (i < 500 && animals[i] != null)
        {
            if (animals[i].GetType().Equals(kind))
                System.out.print(animals[i].GetName() + "," +
                    animals[i].GetNumR() + "," + animals[i].GetAge());
            i++;
        }
    }

    // العملية تعدل بسنة سنوات العمل لكل طبيب
    public void UpdateVetek()
    {
        int i = 0;
        while (i < 10 && veterinarians[i] != null)
        {
            this.veterinarians[i].SetVetek();
            i++;
        }
    }

    // العملية تعدل عمر الحيوان بسنة
    public void UpdateAgeAnimals()
    {
        int i = 0;
        while (i < 50 && animals[i] != null)
        {
            this.animals[i].SetAge();
            i++;
        }
    }
}
```

// العملية تتلقى رقم هوية البيطري وتعيد اسمه  
public String GetVet(int id)  
{  
 int i = 0;  
 while (i < 10 && veterinarians[i] != null)  
 {  
 if (veterinarians[i].GetId() == id)  
 return veterinarians[i].GetName();  
 i++;  
 }  
 return null;  
}

// العملية تتلقى حيوان وبيطري معالج ورقم العلاج وتضيف زيارة  
// مجمع الزيارات لهذا الحيوان  
public void AddAppointment(Animal p, string t, Vet v)  
{  
 int i=0;  
 while (i < 50 && !animals[i].Equals(p))  
 i++ ;  
 Appointment[] a = animals[i].GetLastVisits();  
 i = 0 ;  
 while (a[i] != null)  
 i++;  
 a[i] = new Appointment(v, t);  
}

**سؤال 29 :**

معطى:

```

public class A
{
    private int myVal;
    public A (int val) {myVal = val;}
    public int f ()      {return 1;}
}
public class B extends A
{
    private double x;
    public boolean validCode() { return x > 8.0; }
}

```

(أ)

أمامك أربعة أقوال i-iv . حدّد بالنسبة لكلّ واحد منها إذا كان صحيحاً أم غير صحيح , وعلّل تحديدك .

الفئة A تَرث العملية validCode من الفئة B .

الفئة B تَرث جميع صفات وجميع عمليات الفئة A .

الفئة B يمكنها التوجّه مباشرة إلى صفات الفئة A .

الفئة A يمكنها التوجّه إلى الصفة x للفئة B .

(ب). طبّق في الفئة B عملية بنائية تتلقّى كبارامتر عدداً صحيحاً وعدداً حقيقياً ( بهذا الترتيب ) , وتبتدئ الصفات وفقاً لذلك .

(ج). أمامك تعريفان من البرنامج الرئيسي :

```

A code = new B(127 , 1.4);
A num = new A(613);

```

بالنسبة لكل واحد من الأوامر i-iv التي أمامك، حدّد إذا كاذ سليماً أم غير سليم .  
إذا لم يكن سليماً، علّل تحديدك، واكتب إذا كان الخطأ خطأ تشغيل أم خطأ تجميع.

- i. `boolean myBool = code.validCode();`
- ii. `boolean myBool = num.validCode();`
- iii. `boolean myBool = (B) code.validCode();`
- iv. `boolean myBool = (B) num.validCode();`

(د)  
نريد أن نعرف , أثناء التشغيل , كم مرّة تمّ تشغيل العملية f من كائن من النوع A الذي ليس B, وكم مرّة تمّ تشغيلها من كائن من النوع B .  
إذا كان يمكن الحصول على هذه المعلومة - أضف الأمر اللازم أو الأوامر اللازمة للحصول على المعلومة . انسخ إلى دفترك الفئة أو الفئات التي أضفت إليها الأمر/ الأوامر.

إذا كان لا يمكن الحصول على المعلومة - فسّر لماذا !!

**حل سؤال 29 :**

(أ)

- i. غير صحيح, A هي فئة اساسية. العملية validCode معرفة داخل B.
- ii. صحيح, B ترث A. هي ترث أيضاً كل الميزات وكل العمليات.
- iii. غير صحيح, myVal المعرفة داخل A هي private وليست protected.
- iv. غير صحيح, A لا تعرف ميزات, لذلك لا يمكن ان تتطرق اليهم (حتى لو كانت الميزات خاصة private).

(ب)

```
public B (int val, double x)
{
    super(val);
    this.x = x;
}
```

(ج)

- i. خطأ مصحح (debugger), code هو كائن من نوع B بعد العملية تحول الى الاعلى من نوع A. لذلك مناداة على دالة داخل B غير صحيح.
- ii. خطأ مصحح (debugger), num هو كائن من نوع A لذلك validCode ليست تتبع لهذه الفئة. لذلك مناداة على هذه العملية داخل B غير صحيح.
- iii. خطأ مصحح (debugger), تحويل لنوع B يحدث على نتيجة العملية. العملية تؤدي الى خطأ تصحيح (قسم i), لا يمكن تحويل قيمة منطقية الى متغير منطقي. يجب ان تكون بالصورة :
- iv. boolean myBool = ((B)code).validCode()
- v. خطأ مصحح (debugger), num ليس من نوع A. لذلك لا يمكن ان نحوله الى B. يجب ان تكون بالصورة :
- vi. boolean myBool = ((B) num).validCode()

(د)

المقطع داخل main	تغيير العملية f بالفئة A
<pre>A[] arr = new A[5]; arr[0] = new A (1); arr[1] = new B (2, 2.2); arr[2] = new B (3, 3.3); arr[3] = new A(4); arr[4] = new B (5, 5.5); int countA = 0, countB = 0; for (int i=0 ; i&lt;arr.length ; i++) {     if (arr[i].f() == 1) countB++;     else countA ++; }</pre>	<pre>public int f() {     if (this instanceof B)         return 1;     return 0; }</pre>

**سؤال 30 :**

أمامك مشروع فيه الفئات A و B و Run .

```

public class A
{
    private int n;
    private char ch;

    public A() { n = 2; ch = 'G '; }
    public A(int n) { this.n = n; ch = 'M'; }
    public A(int n , char ch) { this.n = n; this.ch = ch; }
    public A(A other) { n = other.n; ch = other.ch; }
    public int getN() { return n; }
    public char getCh() { return ch; }
    public void inc() { n++; ch++; }
    public String toString()
    {
        String s = "";
        for (int i = 0; i < n; i++)
            s = s + ch;
        return s;
    }
}

//-----
public class B extends A
{
    private A a;

    public B() { super(); a = new A(); }
    public B(int n) { super(n); a = new A(); }
    public B(A other) { super(); a = new A(other); }
    public B(A other , int n) { super(other); a = new A(n); }
    public void inc() { a.inc(); }

    public A makeA()
    {
        return new A(one(this.a.getN(), this.getN()), one(this.a.getCh(),
            this.getCh()));
    }

    private int one(int n, int m)
    {
        if(n > m) return n;
        return m;
    }
}

```

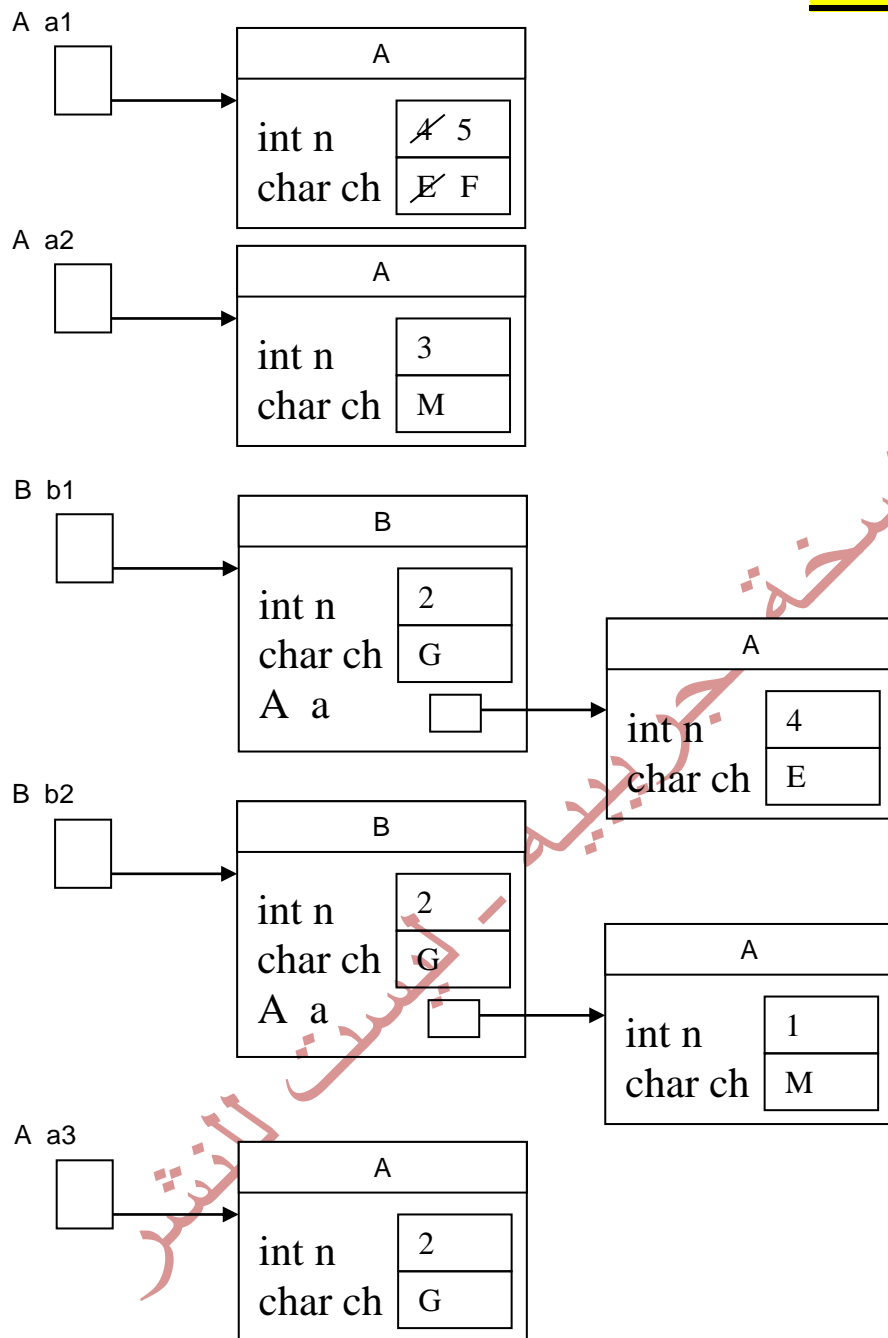
```
private char one(char ch1, char ch2)
{
    if (ch1 < ch2)    return ch1;
    return ch2;
}

public String toString()
{
    return a.toString();
}
}

//-----
public class Run
{
    public static void main()
    {
        A a1 = new A(4 , 'E');
        A a2 = new A(3);
        B b1 = new B(a1);
        a1.inc();
        System.out.println(a1);
        System.out.println(a2);
        System.out.println(b1);
        B b2 = new B(b1, 1);
        System.out.println(b2);
        A a3 = b2.makeA();
        System.out.println(a3);
    }
}
```

اكتب متابعة للعملية main في الفئة Run ، واكتب المخرج .  
يجب أن تكتب في المتابعة قيم المتغيرات، وبالنسبة لكل كائن - قيم صفاته .

## حل سؤال 30 :



المخرج

FFFFF  
MMM  
EEEE  
M  
G G



**سؤال 31 :**

تمتلك مكتبة عامة موقعاً افتراضياً .

يوجد في الموقع 3 أنواع مواد :

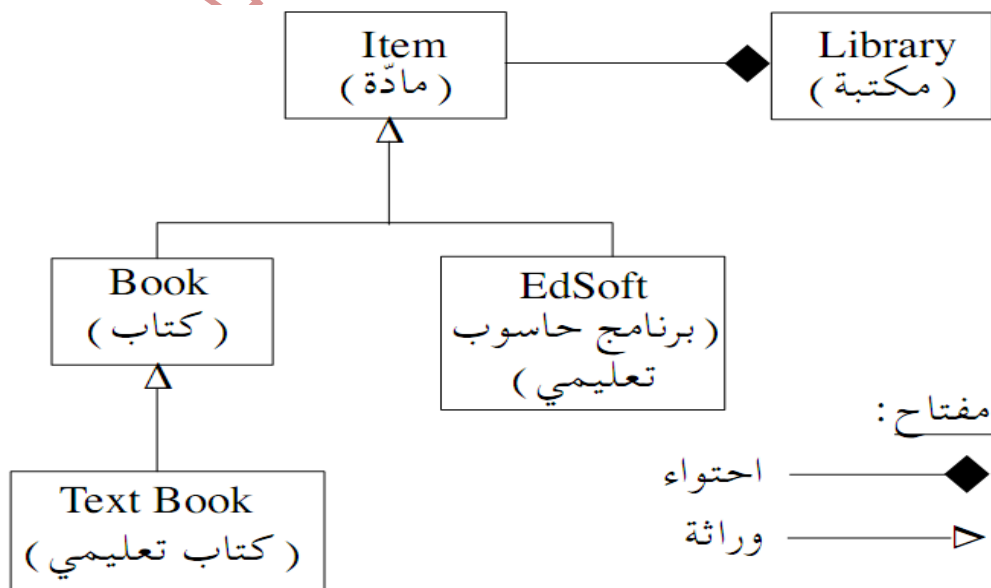
- **كتب :** كل كتاب يشمل كود المادة (الكتاب)، العنوان، اسم المؤلف، عدد الموصيين به.
- **كتب تعليمية :** كل كتاب تعليمي يشمل كود المادة (الكتاب التعليمي)، العنوان، اسم المؤلف، هل الكتاب التعليمي مصادق عليه من وزارة المعارف، عدد الموصيين به.
- **برامج حاسوب تعليمية :** كل برنامج حاسوب تعليمي يشمل كود المادة (برنامج الحاسوب التعليمي)، العنوان، اسم الموضوع، هل برنامج الحاسوب التعليمي مصادق عليه من قبل وزارة المعارف، عدد الموصيين به .

في الكتب والكتب التعليمية العنوان هو اسم الكتاب .  
في برامج الحاسوب التعليمية، العنوان هو اسم البرنامج .

يستطيع المشتركون الدخول إلى موقع المكتبة عن طريق الإنترنت.  
يمكن في الموقع تنفيذ العمليات التالية:

- البحث حسب محدد الموصيين - يُعيد مصفوفة لأكواد المواد التي لها عدد الموصيين المطلوب .
- قراءة كتاب .
- تفعيل برنامج حاسوب تعليمي .
- توصية بعنوان .

أمامك تخطيط لهرمية المغنات بالنسبة للموقع الافتراضي التابع للمكتبة العامة



بالإضافة إلى الفئات الموصوفة في التخطيط , معطاه 3 واجهات تطبيق :

```
interface IReadable {void read(); } // قراءة
interface IRankable {void rank(); } // توصية بعنوان
interface IApprovable {boolean isApproved ();} // مصادق ام لا
```

انسخ إلى دفترك تخطيط هرمية الفئات , وأضف إليه واجهات التطبيق في الأماكن الأكثر ملائمة حسب مبادئ البرمجة الموجهة كائنات .  
استعمل هذه الإشارة ➡ للإشارة إلى تطبيق واجهة تطبيق .

(ب) بالنسبة لكل فئة , أكتب:

عنوانها بلغة Java وصفاتها وعملياتها .

بالنسبة لكل صفة , اكتب تعريفها بلغة Java , وتوثيقها .

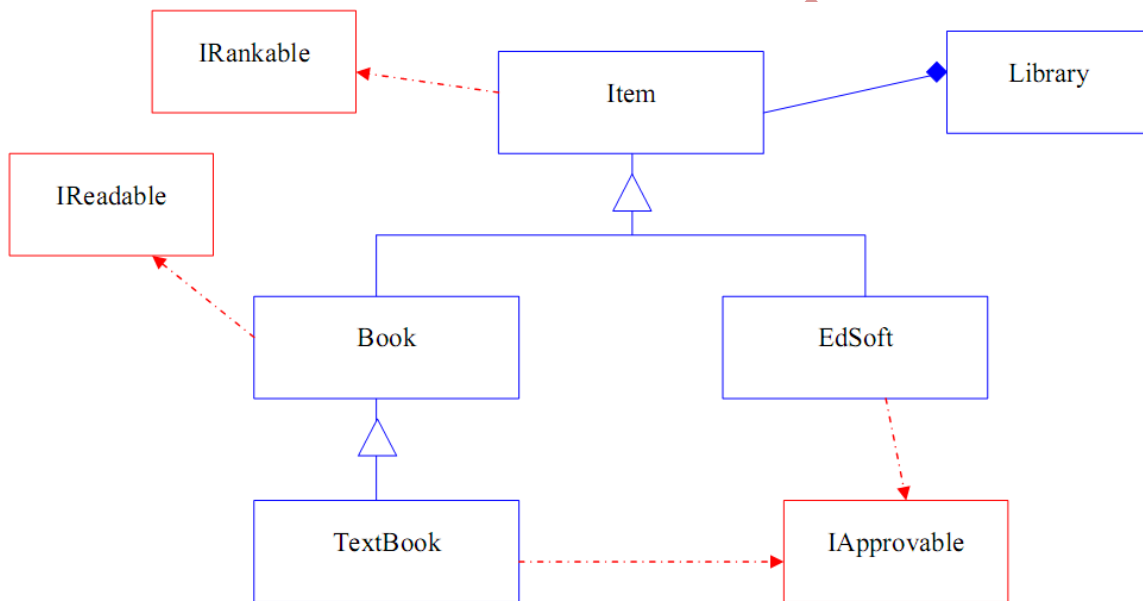
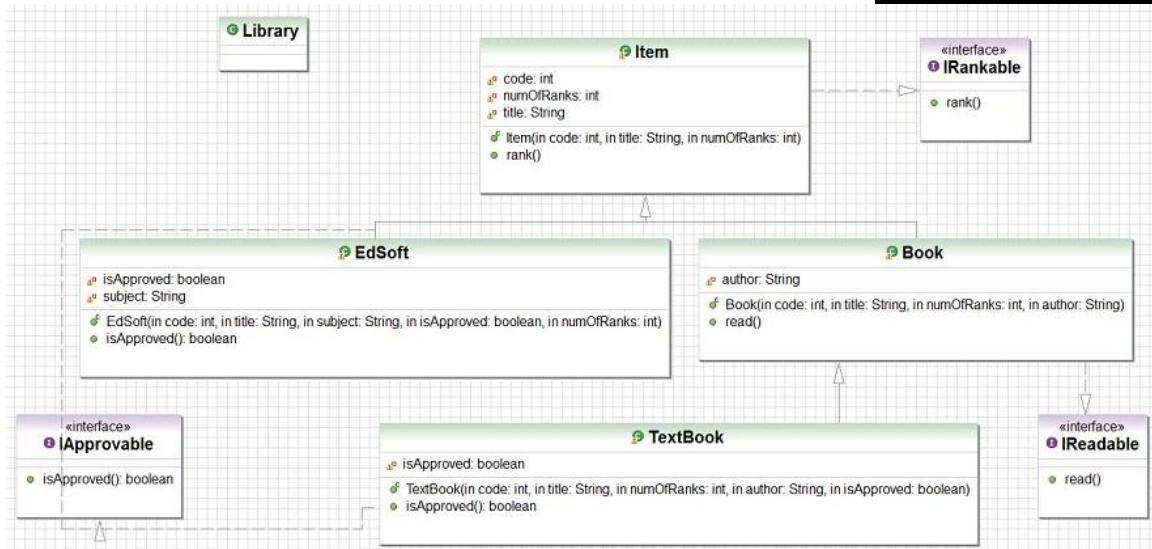
بالنسبة لكل عملية اكتب :

عنوانها بلغة Java وتوثيقاً يشمل ماذا تتلقى وماذا تُعيد . لا حاجة لتطبيق العملية.

لا حاجة لكتابة عمليات بنائية وعمليات محدّدة (عمليات set) وعمليات مُعيدة

(عمليات get) .

## حل سؤال 31 :



```

public class Item implements IRankable
{
    private int code; // رقم الغرض
    private String title; // اسم الكتاب
    private int numOfRanks; // عدد الناصحين

    public Item(int code, String title, int numOfRanks)
    {
        this.code = code;
        this.title = title;
        this.numOfRanks = numOfRanks;
    }

    public void rank() { }
}
  
```

```

public class Library
{
    Item [] arr = new Item[5];
}
  
```

```
public class EdSoft extends Item implements IApprovable
{
    private String subject; // الموضوع
    private boolean isApproved; // هل موافق عليه

    public EdSoft(int code, String title, String subject, boolean isApproved, int numOfRanks)
    {
        super(code, title, numOfRanks);
        this.subject = subject;
        this.isApproved = isApproved;
    }

    public boolean isApproved() { return true; }
}
```

```
public class Book extends Item implements IReadable
{
    private String author; // اسم الكاتب

    public Book(int code, String title, int numOfRanks, String author)
    {
        super(code, title, numOfRanks);
        this.author = author;
    }

    public void read() { }
}
```

```
public class TextBook extends Book implements IApprovable
{
    private boolean isApproved; // هل موافق عليه

    public TextBook(int code, String title, int numOfRanks, String author, boolean isApproved)
    {
        super(code, title, numOfRanks, author);
        this.isApproved = isApproved;
    }

    public boolean isApproved() {return true; }
}
```

**سؤال 32 :**

يعرضون في مركز للفنون ثلاثة أنواع ورشات : ورشة خرزات , رسم على الخشب , رسم على القماش . تُجرى في المركز ورشات سنوية وورشات تشمل 10 لقاءات وورشات لمرة واحدة .

تُجرى الورشات في الصباح وفي المساء . لكل نوع ورشة سعر مختلف . يزود المركز المواد اللازمة لكل ورشة بدون الحاجة لدفع مبلغ إضافي .

الورشات محدودة لعدد أقصى من المشتركين .

يستطيع المشترك أن يسجل لأكثر من ورشة واحدة .

في منظومة إدارة الورشات تُحفظ لكل ورشة التفاصيل التالية : كود الورشة , اسم الورشة , نوع الورشة (خرزات , رسم على الخشب , رسم على القماش) , مدة الورشة (سنوية , 10 لقاءات , لمرة واحدة) , موعد الورشة (في الصباح , في المساء) , سعر الورشة للمشارك , العدد الأقصى للمشاركين , قائمة المشتركين في الورشة.

لكل مشترك يُحفظ اسمه ورقم هاتفه .

يمكن في منظومة إدارة الورشات أيضاً تنفيذ العمليات التالية :

- (1) تسجيل مشترك للورشة .
- (2) فحص إذا كان هناك مكان في الورشة .
- (3) حساب مجموع المدخولات من ورشة واحدة .

أ) ارسم هرمية الفئات بالنسبة لمنظومة إدارة الورشات . اكتب في رسمك بالنسبة لكل فئة اسم الفئة وصفاتها وعناوين العمليات وتوثيقاً للصفات والعمليات . أشر في رسمك إلى الأربطة بين الفئات.

افتراض أنه عُرِّفت لكل صفة عمليات `set/get` , وأنه عُرِّفت لكل فئة عملية بنائية . لا حاجة لكتابة هذه العمليات .

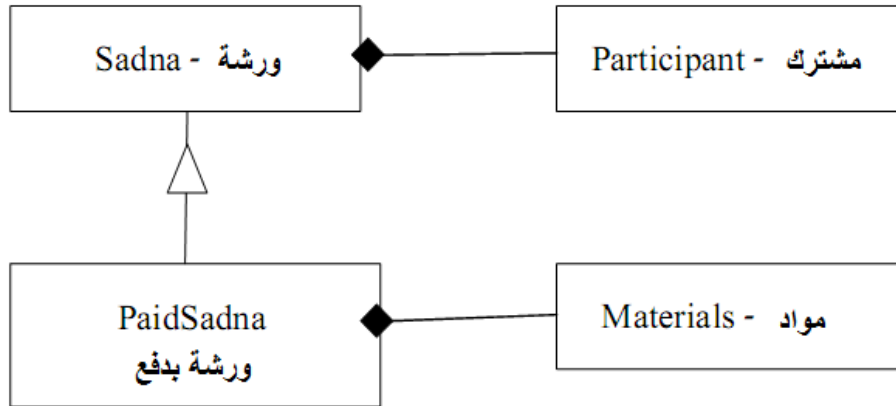
ب) على أثر ارتفاع الأسعار في السوق , تقرر أن لا تُوزَّع الموادّ مجّاناً في قسم من الورشات.

يتوجّب على المشتركين أن يدفعوا مقابل الموادّ , حسب قائمة الموادّ التي سيتمّ تفصيلها أثناء التسجيل للورشة . لكلّ مادّة يحفظ اسمها وسعرها بالنسبة للكمّية اللازمة للورشة . يشمل سعر الورشة السعر الأساسي للورشة وتكلفة الموادّ اللازمة للورشة .

(1) أضف فئات إلى الرسم الذي رسمته في البند "أ" (بدون إضافة أو إنقاص صفات وعمليات في الفئات التي عُرِّفت في البند "أ" ) , بحيث يكون ملائماً لمنظومة إدارة الورشات بعد التغيير , بالطريقة الأكثر ملائمة لمبادئ البرمجة موجهة الكائنات . بالنسبة لكل فئة أضفها , اكتب بلغة Java صفاتها وعملياتها وتوثيقاً للصفات والعمليات .

لا حاجة لكتابة عمليات بنائية وعمليات محدّدة (set) , وعمليات مُعيدة (get).

(2) طبق بلغة Java العمليات التي كتبتها في البند الفرعي "ب 1" .

**حل سؤال 32 :**

```

public class sadna
{
    private int code ;
    private String name;
    private int sug;
    private int duration;
    private char time;
    protected double price;
    private int max;
    private Participant [] arr;
    private int lastPosition;

    public sadna(int code, String name, int sug, int duration,
        char time, double price, int max)
    {
    }

    public void add(Participant p)
    {
    }

    public boolean isFull()
    {
    }

    public double getPrice()
    {
    }

    public int numOfParticipant()
    {
    }

    public String getName()
    {
    }
}
  
```

```
public class PaidSadna extends sadna
{
    private List<Materials> lst;
    private double TotalMatirialPrice;

    public PaidSadna(int code, String name, int sug, int duration,
        char time, int price, int max)
    {
    }

    public void add(Materials mat)
    {
    }

    public double getPrice()
    {
    }
}

public class Participant
{
    private String name;
    private String phone;
}

public class Materials
{
    private String name;
    private double price;
}

public static void main(String[] args)
{
    sadna[] sd = new sadna[3];
    sd[0] = new PaidSadna(123, "Beads - Adults", 1, 2, 'e', 800, 4);

    ((PaidSadna)sd[0]).add(new Materials("Cristal", 34.75));
    ((PaidSadna)sd[0]).add(new Materials("Svaro", 52.50));
    ((PaidSadna)sd[0]).add(new Materials("FireLn", 34));

    sd[0].add(new Participant("noor", "0545-443221"));
    sd[0].add(new Participant("sofia", "0245-443221"));

    sd[1] = new Sadna(234 , "Drawing on wood" , 2,1, 'm' , 50 , 5);
    sd[1].add(new Participant("jmal", "0325-444543"));
    sd[1].add(new Participant("fatima", "0344-466543"));
    sd[1].add(new Participant("fozi", "0655-477883"));
    sd[1].add(new Participant("slima", "0443-993343"));
}
```



1. تقنية البرمجة العليا , المؤسسة العامة للتعليم الفني والتدريب المهني . المملكة العربية السعودية 2010

2. كتاب برمجة كائنات موجهة لايرز كلر (بالعبرية) . 2001

### 3. موقع شركة SUN للغة الجافا .

<http://docs.oracle.com/javase/tutorial/>  
<http://www.oracle.com/technetwork/java/index-jsp-135888.html>

#### 4. موقع شركة netbeans للغة الجافا .

<http://netbeans.org/kb/articles/learn-java.html>

5. برمجة الكائنات الموجهة، الموسوعة العربية للكمبيوتر والانترنت 2002

6. اسس برمجة الكائنات الموجهة، الاستاذ خليل ابو شنب، المكر 2008

7. مواقع باللغة العربية لتعليم لغة الجافا

## ملحق :

لكي تقوم بكتابة برنامج بلغة الجافا ثم تنفيذه لا بد من وجود:  
أولاً : Java 2 Software Development Kit والعروفة اختصاراً بـ JDK أو SDK ، وهي عبارة عن تعليمات اللغة نفسها.

ثانياً : Integrated Development Environment والمعروفة اختصاراً بـ IDE وهي عبارة عن البيئة التي نكتب فيها البرنامج أو المحرر .  
 برنامج eclipse/netbeans هو أحد البرامج التي أنتجتها وطورتها شركة Sun لكي يستخدمه مبرمجو لغة الجافا أي هو IDE .


إذاً ننصب ونحمل برنامج JDK أو SDK أولاً ، ثم بعد ذلك نقوم بتحميل برنامج eclipse/netbeans ، وأثناء عملية التحميل يطلب منا أن نحدد مسار JDK أو SDK.

### بالتفصيل :

تحتاج لتحميل وتثبيت البرامج التالية بالترتيب: (أيضاً لبرمجة الهواتف النقالة)  
jdk7 : يمكن إنزال الملف من الموقع

<http://www.oracle.com/technetwork/java/javase/downloads/java-se-jdk-7-download-432154.html>

حسب نوع ال Windows الذي لديك أو أي برنامج تشغيل ،

 <a href="#">jdk-7-windows-i586.exe</a>	79.48 MB	Windows x86
--	----------	-------------

### netbeans 7.0.1

<http://netbeans.org/downloads/index.html>

عليك تحميل البرنامج تحت النوع: ALL

### أو - eclipse